VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace sledování výroby na dělícím centru

Implementation of workflow on cutting center

2018

Michal Zlacký

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Michal Zlacký**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Implementace sledování výroby na dělícím centru
Implementation of Workflow on Cutting Center

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvoření softwarového balíku pro sledování výroby na dělícím centru v závodě Ferromoravia s.r.o. Staré Město. Obsahem tohoto balíku budou knihovny, sloužící pro komunikaci s PLC zařízeními (SIMATIC), aplikační server a klientská aplikace psaná v jazyce Object Pascal. Komunikace mezi klientem a serverem bude probíhat prostřednictvím protokolu DCOM. Pro přenos dat mezi PLC a serverovou části bude implementována vlastní komunikace.

1. Úvod do problematiky - potřeba sledování výroby, přínosy.
2. Popis použité architektury - vrstvená architektura.
3. Rozbor klientské části - popis protokolu DCOM.
4. Aplikační server - popis komunikace server - aplikační server, zpracování klientských požadavků.
5. Řídící systém - popis zpracování dat z PLC.
6. Online výroba - popis struktur pro rychlé předávání dat - paměťová pole.
7. Perzistence dat - návrh tabulek pro uložení sledování a následné zpracování dalšími aplikacemi.
8. Implementace - klientská aplikace, aplikační server, knihovna pro komunikaci s řídícím systémem.
9. Závěr - zhodnocení výsledku, možnosti rozšíření.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Zbyněk Prokop**

Konzultant diplomové práce: Mgr. Ing. Michal Krumnikl, Ph.D.

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Třinci 30. dubna 2018 ……

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Třinci 30. dubna 2018

Zlacký

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli. Paří mezi ně můj vedoucí práce p.Prokop a také konzultant p.Krumnikl. Bez nich by tato práce nevznikla.

Abstrakt

Cílem této práce je vytvoření softwarového balíku pro sledování výroby na dělícím centru v závodě Ferromoravia a.s. ve Starém Městě. Uvedené centrum slouží k dělení ocelových tyčí na zakázkové délky např. pro automobilový průmysl. Implementace je rozdělena do tří částí: sběr dat, zpracování a vizualizace. Ke sběru dat ze zařízení je vytvořen program komunikující s PLC SIMATIC prostřednictvím protokolu S7. K jejich následnému zpracování slouží programy obsluhující sdílenou paměť. V této paměti se nachází struktury, sloužící k uložení dat o zakázkách, měření či výrobcích. Informace se přenášejí do aplikačního serveru, který je odesílá do klientských stanic. Vizualizační část je napsána v jazyce Object pascal. Umožňuje pracovat s daty z podnikového ERP systému a plánovat jednotlivé zakázky. Operátoři zde mohou sledovat průběh výroby a jsou informováni o aktuálním stavu zařízení.

Klíčová slova: DCOM, SIMATIC, sledování toku materiálu (STM), MES,VIC-25mm

Abstract

The aim of this work is to create a software package for production monitoring in the cutting centre in Ferromoravia Ltd factory in Staré Město. The stated centre divides steel bars into ordered lengths e.g. for a car industry. The implementation is divided into three parts: data collection, processing and visualization. There is a designed programme for collecting the data from the device. This programme communicates with PLC SIMATIC via S7 protocol. There are other programmes using the shared memory, which process these data. In this memory there are the structures which save the data about orders, measurements and products. The information is transferred to the application server, which sends it to the client station. The visualization part is written in Object pascal language. It enables to work with the data from the factory ERP system as well as to plan individual orders. Here the operators can monitor the production process and they are informed about the present device status.

Key Words: DCOM, SIMATIC, STM, MES,VIC-25mm

Obsah

Seznam použitých zkratek a symbolů

DCOM	–	Distributed Component Object Model
PLC	–	Programmable Logic Controller
STM	–	Sledování toku materiálu
ERP	–	Podnikový informační systém
MES	–	Manufacturing Execution Systems
DB	–	Datablok
TŽ	–	Třinecké železárny a.s.
ELVIS	–	Elektronický výrobní informační systém
SŘDB	–	Systém řízení báze dat
Oracle RDB	–	Databázový systém firmy Oracle

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V dnešní době se stále častěji začíná mluvit o tzv. Průmyslu 4.0 nebo také o čtvrté průmyslové revoluci. Tento pojem vychází z potřeby optimalizace výroby, snížení výrobních nákladů a celkového zvýšení efektivity zařízení. Ať už se jedná o úplnou robotizaci nebo pouze částečnou automatizace, zařízení již neslouží pouze k vykonávání určené činnosti. Ve spojení s tzv. nadřazenou úrovní lze k výrobku přiřadit např. konkrétního zákazníka s jeho požadavky. Operátor je neustále informován o parametrech právě zpracovávaného výrobku a je tak schopen okamžitě reagovat v případě nedodržení tolerancí či jiné chyby.

S nadřazenou úrovní neboli také systémem řízení výroby se pojí několik dalších přínosů. Je možné automatizovat opakující se činnosti či odstranit zdvojené agendy. Lze také sledovat chod samotného stroje. Díky této možnosti je způsob, jak eliminovat tzv. předstíranou práci. Operátor se již musí vyjádřit, proč v danou dobu stroj nepracoval. Se sledováním chodu úzce souvisí také optimalizace výroby. Analýzou údajů lze upravit plánování zakázek a tím docílit minimalizace prostojů neboli času, kdy stroj nevyrábí.

Systémy řízení výroby zahrnují procesy plánování, identifikaci a sledování materiálového toku, sběr dat, jejich vizualizaci a v neposlední řadě také řízení kvality. Součástí systémů je obvykle propojení s ERP systémy, správa dokumentů jako jsou informace o průběhu výroby, instrukce pro operátory či receptury pro výrobní zařízení.

Závod Ferromoravia a.s. ve Starém Městě u Uherského Hradiště se zabývá tažením oceli. Od roku 2011 je součástí Třineckých železáren a.s. V roce 2016 tento závod zakoupil a instaloval nové dělící centrum tyčí od izraelské společnosti VIDEX. V této práci bude popsána architektura pro sledování toku materiálu (STM) na tomto centru.

Cílem této práce bylo vytvoření programového balíku určeného ke sledování výroby dělícího centra. Důvodem pro tvorbu tohoto balíku bylo začlenění stroje do informačního systému Třineckých železáren a.s. Pro tyto účely jsem vytvořil klientskou aplikaci a aplikační server psaný v jazyce Object Pascal. Jelikož bylo PLC připojeno k výkonému serveru běžícím na systému OpenVMS, implementoval jsem komunikační programy a datové struktury v jazyce C++.

V úvodní části této práce bude stručně popsán cíl této diplomové práce a její struktura. Na tento úvod naváže kapitola pojednávající o samotném výrobním zařízení. Popíši zde parametry a pracovní režim dělícího centra. Následovat bude popis architektury z pohledu jednotlivých vrstev. V dalších třech kapitolách bude objasněn princip sběru dat z PLC, zpracování těchto dat v řídícím serveru a tvorba vizualizační části. Předposlední kapitola popíše testování jednotlivých komponent. Závěrečná kapitola shrne práci a provede její zhodnocení.

2 Dělící centrum

Dělící centrum se nachází v závodě společnosti FERROMORAVIA, s.r.o. Tento závod se nachází ve Starém Městě u Uherského Hradiště. Hlavním výrobním artiklem je ocel tažená za studena vyráběná buďto ze svitků do tyčí nebo z tyčí do tyčí. Mezi výrobní zařízení patří 5 kombinovaných tažných strojů, 2 tažné stolice a další pomocná zařízení na kontrolu kvality. Závod produkuje cca 90 kt tažené oceli ročně a patří mezi nejvýznamnější výrobce tažené oceli v ČR. V roce 2011 sfúzovala firma s TŽ a.s. a stala se provozem VF - Tažírna oceli TŽ a.s.

Hlavním důvodem pro koupi dělícího centra bylo rozšíření zpracovávaného sortimentu o dělené tyče. Tyto tyče jsou používány v např. v automobilovém průmyslu jako pneumatické držáky zadních dveří. Do investičních akcí byl požadavek na nové dělící centrum zahrnut v roce 2014. Požadavkem bylo dělení materiálu v rozměrech od 8,2 do 30 mm a v délkách 150 - 300 mm s přesností ± 0,3 mm.

Výhercem výběrového řízení se stala izraelská firma VIDEX.

Do závodu bylo dělící centrum instalováno na začátku roku 2016. Jedná se o impaktní řezačku a fasetovací frézu, model VIC-25mm a VCM-25mm-D. Stroj je schopen zpracovávat tažený drát $\phi 8 - \phi 25$, až do pevnosti v tahu 900N/mm^2 pro průměry do 21mm a 700N/mm^2 do průměru 25mm. K tomuto zařízení je připojen podavač tyčí VBF-6000 s rychlostí 50 tyčí za minutu. Celkový pohled je vyobrazen na obr. č.1

Obrázek 1: Impaktní řezačka VIC-25mm

2.1 Technické specifikace

Tento model řezačky má následující specifikace:

Minimální průměr drátu	$\phi 8$
Maximální průměr drátu	$\phi 25$
Délka odřezávání	100 mm - 800 mm, zcela nastavitelné
Opakovatelnost délky	± 0.3 mm, měřeno mezi středy
Výroba	40 obrobků za minutu
Vstupní výška drátů	1254 mm
Vnější výška	800 mm
Hmotnost	4000 kg, včetně hydraulického celku
Rozměr podlahové plochy	2750 mm x 6000 mm
Motor hydraulického čerpadla	30 HP (22 kW); 4 póly, B5
Zapojení ke zdroji	3-fázový, 60 A
Napětí	3-fázový, 400 V, 50 Hz

Tabulka 1: Parametry řezačky

Bylo dohodnuto, že zařízení bude naprogramováno pro řídící systém SIMATIC, aby bylo umožněno propojení s nadřazenou úrovní. Samotný stroj je schopen pracovat i bez přímého spojení s nadřazenou úrovní.

2.2 Pracovní režim stroje

Pracovní postup spočívá ve stříhání jednotlivých tyčí z balíku navezeného na vstupní podavač. Stroj dávkuje tyče po jedné a stříhá dle nastavené délky. Prvních 15 cm tyče je vždy ustřiženo jako technologicky nutný odpad. Toto je zajištěno výsuvnou zarážkou. Po odstřižení začátku tyče, stroj pomocí přítlačných válců posune tyč na doraz, který aktivuje samotný střih. Zpoždění střihu je dáno nastavením stroje. Počet střihů, které budou provedeny záleží na délce vstupní tyče. Ta je také zadána jako parametr stroje. Na základě této délky je vypočten potřebný počet střihů a výsledný odpad z každé tyče. Je žádoucí, aby tento odpad byl co nejmenší, proto se pracovníci provozu snaží zajistit vstupní materiál vhodné délky.

Ustřižená tyč putuje buďto do levého zásobníku, ve kterém se nachází odstřižené začátky a konce tyčí nebo doprava. Zde se řadí do tzv. bufferu, který tyče dávkuje k frézám. Frézování probíhá uchopením tyčky a následným ofrézováním jednoho či obou konců. Po ofrézování je tyč položena zpět na kluznou plochu, po které pokračuje do kapsy. Z tohoto zásobníku je pracovníkem vyjmuta a vložena do připravené bedny. Počet tyčí je definován velikostí bedny. Uložení jednotlivých kusů si volí zákazník.

Po naplnění celé bedny je na povel operátora vytištěn expediční štítek. Na něm se nacházejí veškeré informace o zakázce, jako je tavba, délka výrobku s tolerančními mezemi, počet kusů v bedně, teoretická hmotnost vypočítaná na základě délky a průměru výrobku a také identifikační číslo samotné bedny, které slouží k přesné identifikaci a archivaci záznamu.

Rychlost stříhání je závislá na délce, průměru a jakosti vstupního materiálu. Výrobcem stanovený průměrný výkon je 40 střihů za minutu, což odpovídá času naplnění jedné 200 kusové bedny za 5 minut. Tento čas však v reálných podmínkám nelze dosáhnout, neboť je nutné připočíst dobu dávkování vstupního materiálu.

Pro kontrolu střižené délky bylo do stroje instalováno laserové měření délky, které v reálném čase vyhodnocuje správnost střihu a zajišťuje, že se do výstupní bedny nedostane tyč nesprávné délky. Měření probíhá před frézováním a na základě vyhodnocení správnosti je tyč vyhozena jako vadná nebo pokračuje směrem k frézám.

O řízení stroje se stará programovatelný automat Siemens SIMATIC S7-300. Komunikace s tímto automatem je předmětem této práce a bude popsána níže.

2.3 Nastavení stroje

Nastavení stroje lze upravit pro konkrétní požadavky zákazníka. Hlavním parametrem je požadovaná délky výrobku. Ta se zadává prostřednictvím OP (operačního panelu). Mezi další potřebné parametry patří délka vstupní tyče, zpoždění a doba trvání střihu, prodleva vyhazovače materiálu nebo například zapnutí čí vypnutí vzduchového čištění fréz. Ukázka nastavení je na obr. č.2

Obrázek 2: Nastavení stroje: Vlevo časy trvání a zpoždění střihu, vpravo zbytková délka, počet výrobků na kus a celkový počet

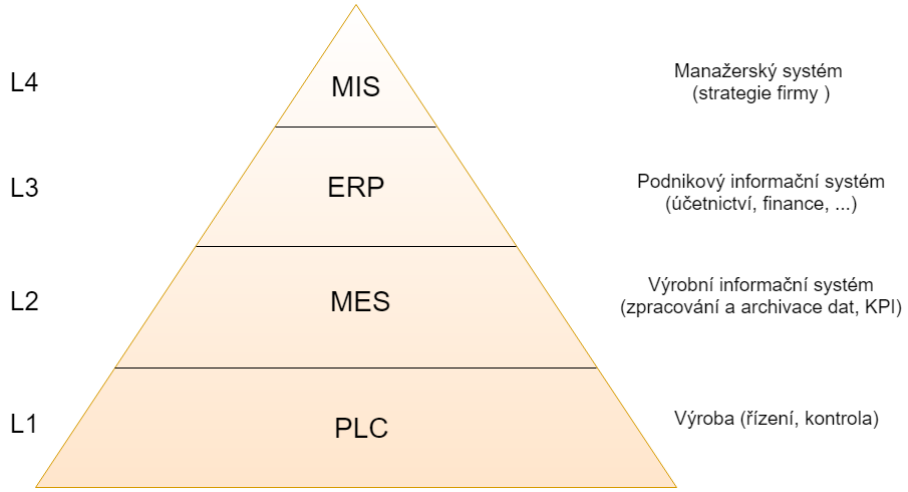
2.4 Integrace zařízení do výrobního procesu

Jak již bylo uvedeno výše, zařízení dokáže pracovat autonomně. Avšak z důvodu integrace zařízení do stávajícího systému, bylo rozhodnuto, napojit dělící centrum do již zavedeného informačního systému. Tento proces zahrnoval vytvoření kompletní třívrstvé architektury počínaje sběrem dat z PLC, přes zpracování dat na hlavním řídícím serveru až po tvorbu samostatné klientské aplikace pro vizualizace dat. Tento krok také umožnil propojit zařízení s ERP systémem společnosti.

V následující kapitole bude architektura užitá v této práci důkladněji popsána.

3 Popis architektury

Sledování toku materiálu (STM) je proces, jenž spadá do oblasti tzv. MES (Manufacturing Execution System) systémů. Tyto systémy slouží k získávání a zpracování dat z výrobních zařízení v reálném čase, prostřednictvím komunikace s PLC automaty. Na obr č.3 je znázorněna pyramida informačních systému, dělící tyto systémy do různých úrovní.



Obrázek 3: Rozložení vrstev informačních systémů

Na nejnižší úrovni se nacházejí právě PLC systémy. Periferie těchto systému jsou napojena přímo na výrobní zařízení a umožňují řízení v reálném čase.

Nad úrovní PLC se nachází MES úroveň, o které je tato práce. MES systémy se vyznačují velkou variabilitou vzhledem k výrobním požadavkům. Velmi zastoupeny jsou např v automobilovém průmyslu nebo právě v hutním odvětví. Třinecké železárny a.s. využívají podnikový informační systém ELVIS (Elektronický výrobní informační systém).

O úroveň výše se nachází účetní systémy, systémy plánování a lidských zdrojů.

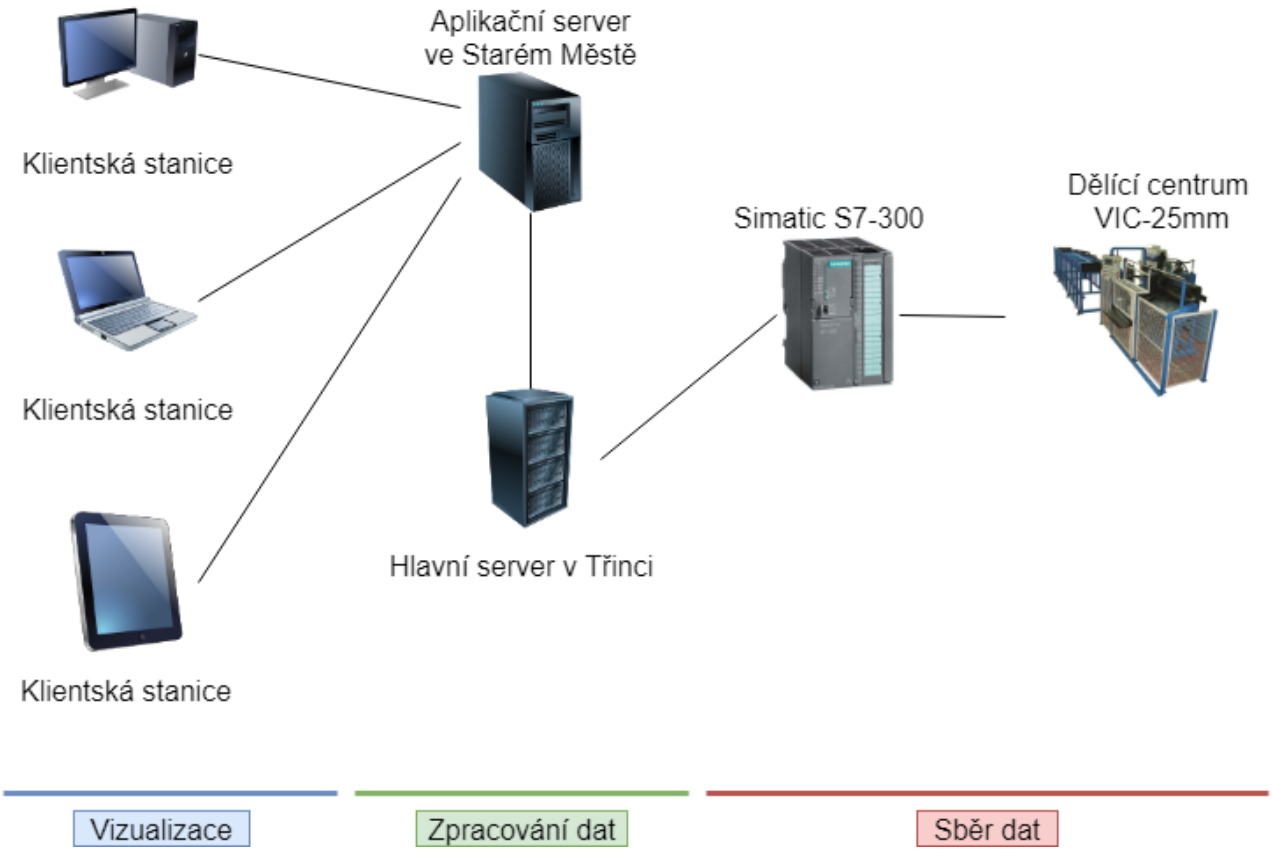
Na nejvyšší úrovni jsou MIS neboli manažerské informační systémy, které se zabývají strategickým plánováním a rozvojem, marketingem apod.

3.1 Architektura

Základem architektury použité v této práci je model CLIENT-SERVER. Tento model umožňuje práci se strojem na více pracovištích a zároveň centralizuje potřeby nejrůznějších ovladačů a nástrojů ke komunikaci se zařízením.

Ukázka architektury použité v této práci je na obr. č.4

Na obrázku je patrné propojení různých klientských zařízení s aplikačním serverem. Tento server je umístěn v závodu ve Starém Městě a má na starosti obsluhu klientských požadavků a komunikaci s hlavním serverem. Hlavní server se nachází v Třinci a slouží ke komunikaci s řídícím automatem. Oba servery mohou také komunikovat s databází.



Obrázek 4: Ukázka architektury CLIENT-SERVER použité v rámci diplomové práce

V případě klientských zařízení se jedná o klasické stolní počítače, na kterých běží tlustý klient. Aplikační server je výkonný počítač, na kterém se nachází serverová část aplikace a ovladače pro komunikaci s databází. Jako hlavní server je použita stanice společnosti HP se systémem OpenVMS. Ta se stará o přenos dat mezi řídícím automatem a aplikačním serverem.

Z rozdělení architektury vznikly tři samostatné úlohy, které bylo třeba řešit.

Prvním úkolem byl samotný sběr dat z PLC do hlavního serveru. Programovatelný automat komunikuje prostřednictvím protokolu S7, proto bylo třeba zajistit komunikaci přes tento protokol.

Druhá část se týkala zpracování dat do podoby vhodné k ukládání a vizualizaci. Implementoval jsem třídy pro lepší práci s těmito daty a tabulky pro jejich archivaci. Následně bylo třeba vyřešit přenos dat na aplikační server. Mezi servery probíhá komunikace formou výměny zpráv na protokolu TCP-IP.

Posledním úkolem byla implementace komunikačního rozhraní mezi klientskou aplikací a aplikačním serverem. Zde jsem využit protokolu DCOM. Samotná aplikace byla vytvořena v jazyce Object Pascal.

V následujících kapitolách budou popsány jednotlivé části týkající se sběru, zpracování a vizualizace dat z dělícího centra.

4 Sběr dat

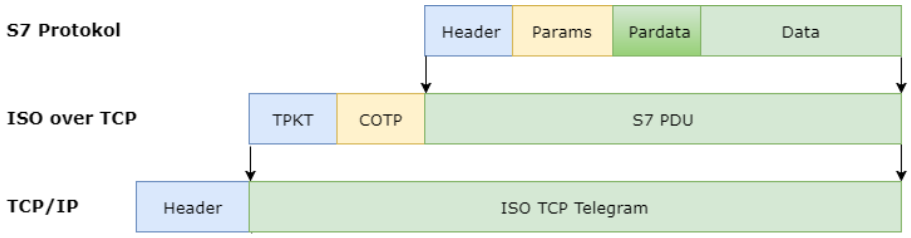
V této kapitole se zaměřím na sběr dat a technologických parametrů zařízení. Jedná se o zprostředkování komunikace mezi PLC a řídícím serverem. Ve společnosti Třineckých železáren se využívají dva výhradní dodavatelé těchto automatů. Jedním z nich je společnost ABB, druhým společnost Siemens. Ke sledování dělícího centra byl jako PLC vybrán typ SIMATIC S7-300 společnosti Siemens.

Simatic S7-300 je programovatelný logický ovladač (PLC) společnosti Siemens. Stará se o zpracování signálů ze stroje a řízení. Je dodáván v šesti výkonnostních variantách. Komunikace s tímto PLC je zprostředkována prostřednictvím protokolu S7.

4.1 Protokol S7

Protokol S7 je hlavním protokolem komunikace zařízení Siemens. Siemens používá tradiční MASTER-SLAVE nebo CLIENT-SERVER model, kde PC (MASTER nebo CLIENT) odesílá požadavky do zařízení (SLAVE nebo SERVER). Tyto informace jsou použity na získání nebo vložení dat do zařízení nebo provedení určitého příkazu.

Ethernetová varianta protokolu S7 využívá blokově orientované ISO transportní služby. S7 protokol je zapouzdřen do TPKT a ISO-COTP protokolu, který umožňuje přenos PDU (Protokol Data Unit) skrz TCP. Komunikace ISO over TCP je definována v RFC1006[11], ISO-COTP je zadefinována v RFC2126[12], který je založen na ISO 8073 protokolu (RFC905[13]). Na obrázku č.5 je vidět zapouzdření protokolu S7 do TCP/IP paketu.



Obrázek 5: Zapouzdření protokolů

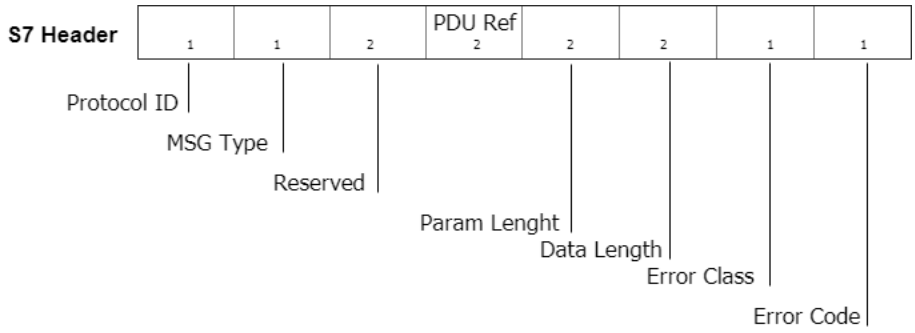
Protokol S7 je typu příkaz/funkce, což znamená, že přenos se sestává z požadavku následovaného příslušnou odpovědí. Počet souběžných přenosů a maximální délka zprávy je určena během nastavení přenosu.

PDU protokolu S7 se skládá ze tří částí:

- Hlavička: obsahuje informace o délce zprávy, odkaz na PDU a konstantu určující typ zprávy.
- Parametry: obsah a struktura se velmi liší v závislosti na zprávě a typu funkce PDU
- Data: toto volitelné pole nese konkrétní informace např. paměťové hodnoty, databloky, firmware atp.

4.1.1 Hlavička protokolu

Hlavička protokolu je 10-12 bytů dlouhá, zprávy obsahující odpověď skládající se ze 2 bytů navíc. Tyto byty jsou určeny pro chybový kód. Formát hlavičky je stejný napříč všemi PDU. Diagram zobrazující vzhled zprávy je na obr. č.6



Obrázek 6: Hlavička protokolu S7

Popis jednotlivých polí:

- Protocol ID:[1B] konstanta protokolu, vždy nastavena na hodnotu 0x32
- Message Type:[1B] základní typ zprávy (někdy označována také jako ROSCTR typ)
  - 0x01 - Job Request: požadavek odeslaný masterem (např. čtení/zápis do paměti, čtení/zápis bloků, start/stop zařízení atp.)
  - 0x02 - Ack: jednoduché potvrzení, odeslaná slavem bez jakýchkoli dat
  - 0x03 - Ack-Data: potvrzení s dodatečnými daty, obsahuje odpověď na job request.
  - 0x07 - Userdata: rozšíření původního protokolu, pole parametrů obsahuje ID požadavku či odezvy (užívané pro programovaní či ladění, funkce zabezpečení, nastavení času,...)
- Reserved:[2B] vždy nastaveny na 0x0000
- PDU Reference:[2B] je generována masterem, inkrementována z každým novým přenosem, užívána ke spárování požadavku s příslušnou odezvou.
- Parameter Length:[2B] délka pole paremetrů, kódování Big-Endian
- Data Length:[2B] délka datového pole, kódování Big-Endian
- (Error class):[1B] je přítomna pouze u Ack-Data zpráv
- (Error code):[1B] je přítomna pouze u Ack-Data zpráv

Zbytek zprávy velmi závisí na jejím typu.

4.1.2 Data

Datová část PDU protokolu S7 se mění v závislosti na typu(čtení/zápis) a směru zprávy.

Sestává se z následujících částí:

1. Read Request: datová část je prázdná
2. Read Response: datová část Act-Data zprávy obsahuje strukturu datových položek, jednu pro každou Request Item nacházející se v původním požadavku. Tyto položky obsahují aktuální hodnoty proměnných.
3. Write Request: obsahuje podobné datové položky jako Read Response, jednu pro každou Request Item uvedenou v poli parametrů hlavičky.
4. Write Response: datová část zprávy obsahuje 1 byte chybového kódu pro každou Request Item z původního Write Request.

Reqest Item vždy obsahuje popis proměnných, zatímco datové položky udávají jejich hodnoty. Struktura datových položek musí začínat na sudém bytu. Jestliže je jejich délka liché číslo, pak za každou takovouto položkou následuje nula.

Typy proměnných u datových položek mohou nabývat následující hodnot:

1. BIT:[X] - jeden bit
2. WORD: dva byty bez znaménkového celého čísla
3. DWORD: 4 byty znaménkového celého čísla
4. REAL: 4 byty čísla s plovoucí řadovou čárkou

Například adresa proměnné DB100X2.1 značí druhý bit třetího bytu databloku 100.

Diagram výměny dat je zobrazen na obr č.7

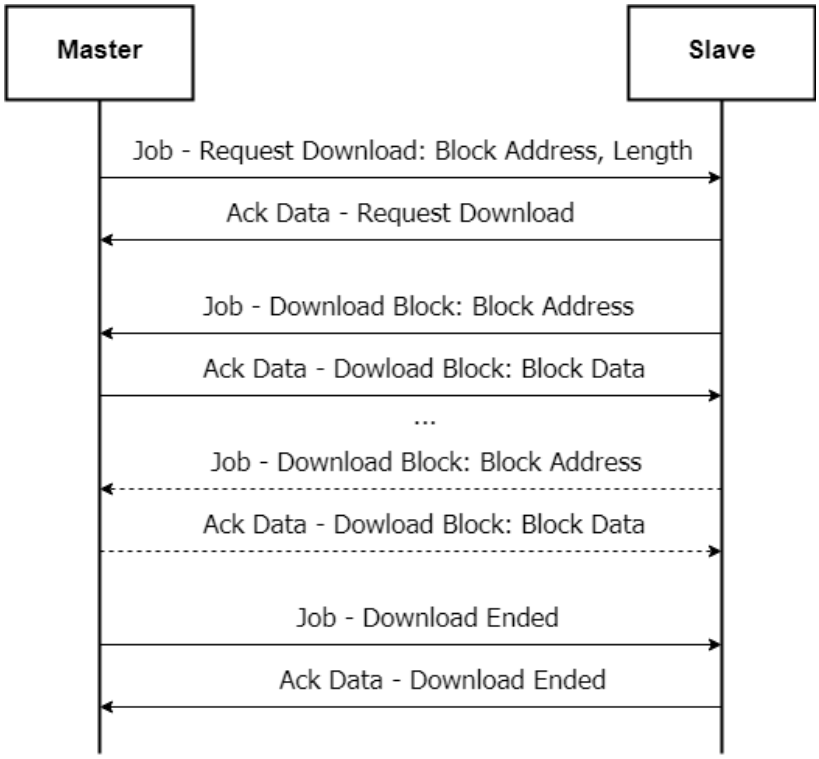
4.2 Příklady komunikace s PLC Simatic

V současné době se na internetu nachází několik knihoven pro různé platformy, sloužící ke komunikaci s PLC Simatic. Patří mezi ně např. SNAP7, libnodave, PLCcom for S7 a další. Tyto knihovny jsou k dispozici pro systémy Windows, Linux, případně ARM.

Knihovny SNAP7 [4] a libnodave [6] jsou distribuovány pod veřejnou licencí, konkrétně LGPLv3 a LGPLv2, umožňující užití v komerční sféře.

PLCcom for S7 [7] je součástí softwarového balíku STEP 7 Professional společnosti Siemens. Knihovna může být také distribuována samostatně. Společnost Siemens užívá vlastní licenční politiku definovanou několika licenčními typy. Tyto typy se liší dobou platnosti, počtem PC i umístěním licenčního klíče.

Příklad načtení databloku prostřednictvím knihovny SNAP7 je na výpisu č.1



Obrázek 7: Diagram stažení databloku protokolu S7

```
Uses Snap7;

Var
MyDB32 : packed array[0..255] of byte; // generic buffer
MyClient : TS7Client;

Procedure SimplyGet;
Begin
 MyClient:=TS7Client.Create;
 MyClient.ConnectTo('192.168.10.100',0,2);
 MyClient.DBRead(32,     // DB Number
 0,       // Start from
 16,       // How many
 @MyDB32); // Target address
 MyClient.Free;
End;
```

Výpis 1: Ukázka práce s knihovnou SNAP7

Na tomto výpisu je vidět volání konstruktoru třídy *TS7Client*. Tato třída se stará o navázání spojení s PLC a následné vyčítání potřebných dat. Ke spojení slouží metoda *ConnectTo()*. K samotnému čtení je určena metoda *DBRead()*, jejímiž parametry jsou číslo databloku, offset, počet bytů a adresa bufferu, do kterého se data uloží.

V této práci jsem z důvodu užití operačního systému OpenVMS nebyl schopen využít žádnou z výše zmiňovaných knihoven. Ke komunikaci mi sloužila knihovna napsaná v jazyce C++, jejíž vlastníkem je firma Danieli CEDA S.p.A. Tato knihovna byla zakoupena společně s výrobním zařízením instalovaným do Třineckých železáren a.s.

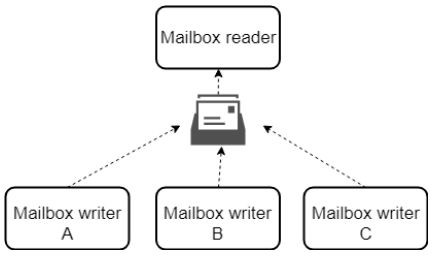
4.2.1 Procesy a databloky pro dělící centrum

Pro komunikaci se řídícím automatem SIMATIC jsem vytvořil proces (výpis č.3), který prostřednictvím knihovny komunikuje s PLC. Jeho účelem je číst a zapisovat databloky, ve kterých se nachází veškeré informace k řízení stroje. Po načtení je datablok vložen do mailboxu, ze kterého je následně vyčítán procesem, starajícím se o zápis do sdílené paměti.

4.2.2 Systém mailboxů

Systém OpenVMS nabízí několik metod pro mezi-procesovou synchronizaci a komunikaci [10]. Mezi tyto metody patří např, sdílené soubory, logická jména, mailboxy a globální sekce. Pro mé účely jsem zvolil systém mailboxů.

Mailboxy jsou pseudo-zařízení, podobná pípám na systémech UNIX. Oproti nim však umožňují obousměrnou komunikaci, tzn. že jeden proces může číst a zapisovat do stejného mailboxu. Zprávy se řadí do fronty systémem FIFO (First-IN-First-OUT). Každému mailboxu lze přiřadit kanál a tím jej omezit na pouze čtecí či pouze zapisovací mailbox.



Obrázek 8: Ukázka zápisu a čtení mailboxů

Než může být mailbox použit, musí být vytvořen. Tvorba probíhá voláním systémové funkce *sys$crembx*. Jednotlivými parametry nastavujeme, zda se jedná o dočasný či permanentní mailbox, číslo kanálu, velikost zprávy, celková velikost mailboxu, logické jméno a mód. Velikost mailboxu je systémem omezena na 65kb. Zápis a čtení umožňuje metoda *sys$qio*. Dočasný mailbox zaniká, ne-li k němu přiřazen žádný kanál. Permanentní mailbox musí být vymazán explicitně.

V této práci byly použity výhradně permanentní mailboxy.

Ukázka jednoduchého zápisu do mailboxu je na výpisu č.2. Program vyčítá řádky ze standardního vstupu a odesílá je do mailboxu.

```
/* Create/assign a channel to the data mailbox. */
status = sys$crembx(0,&chan,MAX_MSG,BUF_QUO,MBX_PROT,0,&mbx,0,0);
check(status);
/* Get an available event flag number. */
status = lib$get_ef(&efn);
check(status);
/* Read from standard input and send to mailbox until EOF. */
while(gets(in_buffer)) {
 /* If input buffer is too large, abort. */
 if(strlen(in_buffer)>MAX_MSG) {
  sys$exit(SS$_BUFFEROVF);
}
status = sys$qiow(efn,chan,IO$_WRITEVBLK,&ios,0,0,in_buffer,strlen(in_buffer)
    ,0,0,0,0);
 check(status);
 check(ios.iosb$w_status);

/* Send an EOF to the mailbox. */
status = sys$qiow(efn,chan,IO$_WRITEOF,&ios,0,0,0,0,0,0,0,0);
```

Výpis 2: Ukázka zápisu do mailboxu

4.2.3 Proces ComDC1

Ke komunikaci s PLC jsem vytvořil proces ComDC1. Proces ve dvou nekonečných smyčkách přistupuje k PLC a v sekundových intervalech vyčítá a zapisuje databloky. Pokud nastane chyba v některém z uvedených kroků, provede se odpojení a po 10 sekundách je pokus opakován. Po úspěšném čtení je datablok zapsán do mailboxu, na který je napojen proces SimatMgr. Tento proces bude popsán v kapitole Zpracování dat.

```
FOR(ever)
   inicializuj spojeni s PLC
   FOR(ever)
    FOR X=1 TO pocet ctecich databloku
      IF NOT nacti data z databloku do mailboxu THEN break;
    ENDFOR
    cekej 1 sekundu
    FOR X=1 TO pocet zapisovacich databloku
```

```
      IF NOT zapis data do PLC THEN break;
    ENDFOR
    cekej 1 sekundu
   ENDFOR
ENDFOR
odpoj se od PLC
cekej 10 sekund
```

Výpis 3: Pseudokód procesu COMDC1

4.2.4 Databloky

Na základě parametrů stroje a požadavků provozu byly vytvořeny dva čtecí a jeden zapisovací datablok. První čtecí datablok obsahuje informace o chodu stroje, údaje z čidel a nastavení. Ve druhém se nachází aktuální změřená hodnota z laserového měření. Tento datablok je vytvořen formou kruhového bufferu o 60 hodnotách, aby v případě výpadku bylo možné dočíst hodnoty zpětně.

Zapisovací datablok slouží k zastavování stroje z důvodu překročení parametrů zakázky a také k nastavení časů laserového měření.

4.2.4.1 DB18 Datablok DB18 je slouží jako hlavní čtecí datablok. Jeho velikost je 486 bytů. Obsahuje nekonečné čítače čidel, bitové informace aktuálního stavu a informace z ovládacího panelu dělícího centra. Na obr č.9 je znázorněna část databloku zobrazující adresy čítačů střihu, fréz a některých bitových stavů.

Name	Data type	Offset	Start value	Retain	Accessible from HMI	Visible in HMI	Setpoint	Comment
Static								
KomCitac	Int	0.0	-22987	False	True	True	False	Čítač pro kontrolu komunikace
StrihCitac	UDInt	2.0	5823	False	True	True	False	Čítač střihů
StrihDobryCitac	UDInt	6.0	3367	False	True	True	False	Čítač střihů, dobrý kus
StrihSpatnyCitac	UDInt	10.0	416	False	True	True	False	Čítač střihů, odpadní kus
TycCitac	UDInt	14.0	420	False	True	True	False	Čítač tyčí do stroje
KusFrezaCitac	UDInt	18.0	3381	False	True	True	False	Čítač ofrezovaných kusů
KusFrezaCitacDC1	UDInt	22.0	0	False	True	True	False	Čítač dobrých kusů na výstupu z linky
ReservaCitac	Array[1..6] of UDInt	26.0		False	True	True	False	Čítač, rezerva
ReservaCitac[1]	UDInt	0.0	0	False	True	True	False	
ReservaCitac[2]	UDInt	4.0	0	False	True	True	False	
ReservaCitac[3]	UDInt	8.0	0	False	True	True	False	
ReservaCitac[4]	UDInt	12.0	0	False	True	True	False	
ReservaCitac[5]	UDInt	16.0	0	False	True	True	False	
ReservaCitac[6]	UDInt	20.0	0	False	True	True	False	
RESIDE_SENSOR	Bool	50.0	TRUE	False	True	True	False	I0.0
FEED_SENSOR	Bool	50.1	TRUE	False	True	True	False	I0.1
AIR_PRESSURE	Bool	50.2	TRUE	False	True	True	False	I0.2

Obrázek 9: Zkrácený výpis čtecího databloku DB18 dělícího centra

5 Zpracování dat

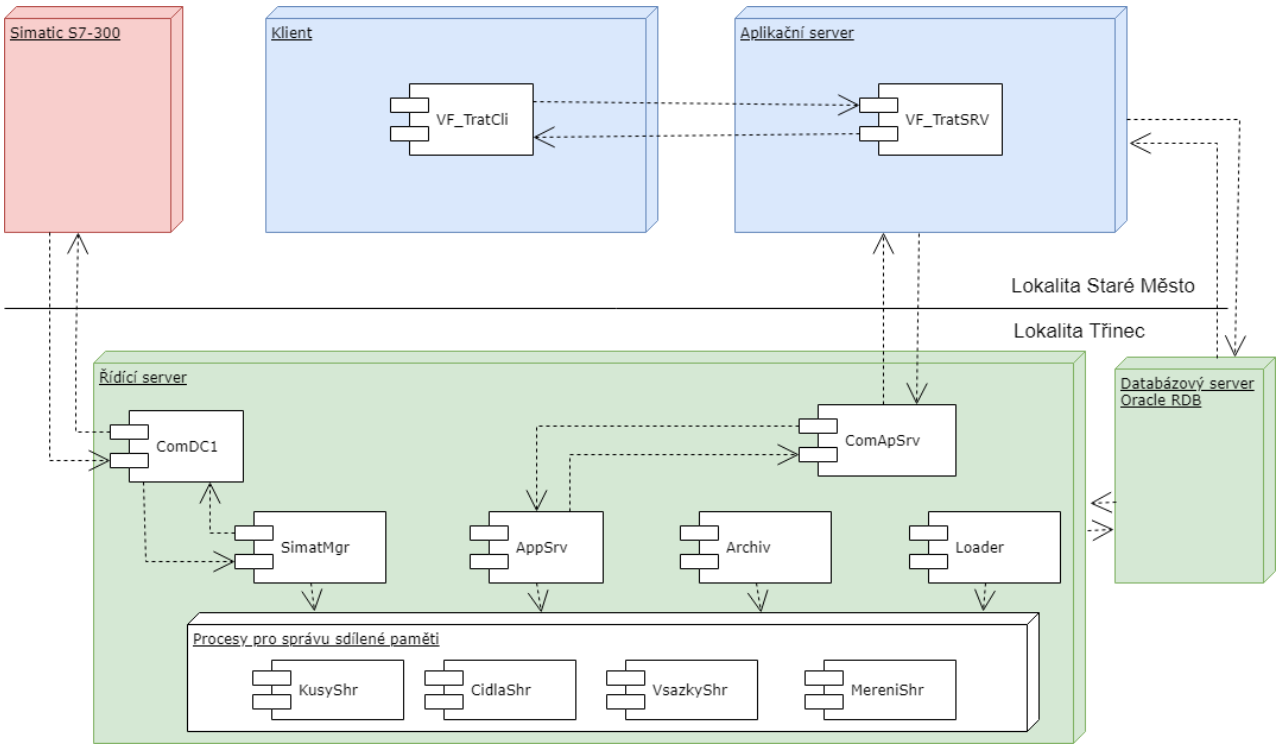
V této kapitole popíši práci s daty, která byla získána z PLC či od uživatele. Vzhledem k rychlým dějům byly použity paměťové struktury umožňující minimalizovat prodlevy při zpracovávání dat. Součástí této kapitoly je návrh datového modelu, který slouží k archivaci dat. Třinecké železárny a.s. využívají SŘDB Oracle RDB, pro který jsem tento model navrhoval.

Na obr. č.10 lze vidět procesní schéma zpracování informací z PLC. Jsou na něm zakresleny procesy (spustitelné soubory) a komunikace mezi nimi.

Sběr dat probíhá procesem ComDC1, který byl popsán výše. Na tento proces je napojen program SimatMgr, který data z PLC odchytává a ukládá je do sdílené paměti. Tuto paměť obsluhují speciální třídy pro kusy, zakázky, měření a čidla. Z této paměti si data mohou vyčítat archivační a nahrávací služby. K archivaci jsem vytvořil proces Archiv, pro plnění struktur z databáze slouží proces Loader.

Pro klientskou aplikaci jsou data přenášena skrze aplikační server prostřednictvím procesů AppSrv a ComApp. První z procesů se stará o přístup ke sdílené paměti, druhý řídí samotný přenos.

Každý z procesů bude níže popsán.



Obrázek 10: Procesní schéma STM Dělícího centra

5.1 Paměťová pole

Dělení materiálu, resp. práce většiny výrobních zařízení, je rychlý proces. Sledování zahrnuje zaznamenávání veškerých změn a jejich kontrola. Pokud jsou v požadavcích zakázky uvedeny konkrétní počty kusů do beden, je nutné tento parametr dodržet a zastavit stroj v případě naplnění bedny. Obsluha poté pouze vymění plnou bednu za prázdnou a proces může pokračovat.

Druhá věc, která se často děje, je interakce uživatele se zařízením. Pokud se uživateli zastaví stroj, měl by být okamžitě informován, z jakého důvodu se to stalo. Uživatel by si v takovémto případě neměl všimnout nějakého zpoždění. Aby bylo možné dosáhnout této míry interakce, je vhodné si objekty držet v paměti.

Dalším důvodem pro užití takovýchto struktur je mezi-procesová komunikace. Každý proces má možnost přistoupit do stejné paměťové oblasti a pracovat se stejnými daty.

Většina moderních operačních systému podporuje určitou formu paměťově mapovaných polí. Systémy jako jsou UNIX, Linux nebo OpenVMS, jsou díky POSIX specifikaci vybaveny mechanismem pro tvorbu takovýchto polí. V systému Windows je možné využít API funkce pro tento účel, např. *CreateFileMapping()*[8]

Jak již bylo zmíněno výše, v systému OpenVMS se k mezi-procesové komunikaci používá více způsobů. Jedním z nich je systém mailboxů, který byl popsán výše. Druhý způsob je systém sdílených souborů, ke kterým lze přistupovat z různých procesů. OpenVMS umožňuje procesu nastavit parametry pro sdílení.

5.2 Zpracování a ukládání dat do sdílené paměti

Jakmile jsou data získána z PLC je potřeba je uchovat. Pro tento účel jsem vytvořil struktury a třídy pro práci s nimi. Základním požadavkem provozu bylo sledování zpracování vstupních svazků, stříhaných tyčí a jejich správné zařazení dle zakázek. Vzhledem k integraci laserového měření požadoval provoz také kontrolu a záznam délky jednotlivých tyčí.

Struktury, které jsem na základě těchto požadavků vytvořil, jsou následující:

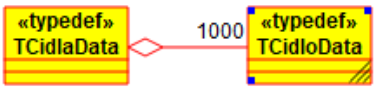
- TKusData - struktura obsahují informace o jednom svazku či bedně.
- TTycData - tato struktura obsahuje informace o jednotlivých tyčích. Je určena pro detailnější sledování.
- TVsazka - v této struktuře se nachází informace o zakázce.
- TCidloData - zde se nachází informace z čidel. Z těchto informací se následně vytváří kusy či tyče.
- TItemMereni - obsahuje záznam o měření.
- TStroj - obsahuje aktuální data ze stroje
- TPracoviste - v této struktuře se nachází informace o přihlášených uživatelích

Tyto struktury jsou uloženy ve sdílené paměti a přístup k nim je pomocí objektů pro práci s nimi. Struktury mají pevně definovanou velikost danou počtem objektů. Počet objektů je zvolen vzhledem k počtu zařízení, která k nim přistupují. Aktuálně s těmito strukturami pracuje 10 zařízení, mezi něž patří sedm tažných strojů, dělící centrum a dvě defektoskopické linky.

Výhodou takovéhoto řešení je okamžitý přístup ke všem datům, neboť se tato data nacházejí v rychlé paměti. Nevýhodou je jejich ztráta v případě restartu serveru či ztráty napájení. Proto se struktury průběžně ukládají na disk ve formě binárních souborů.

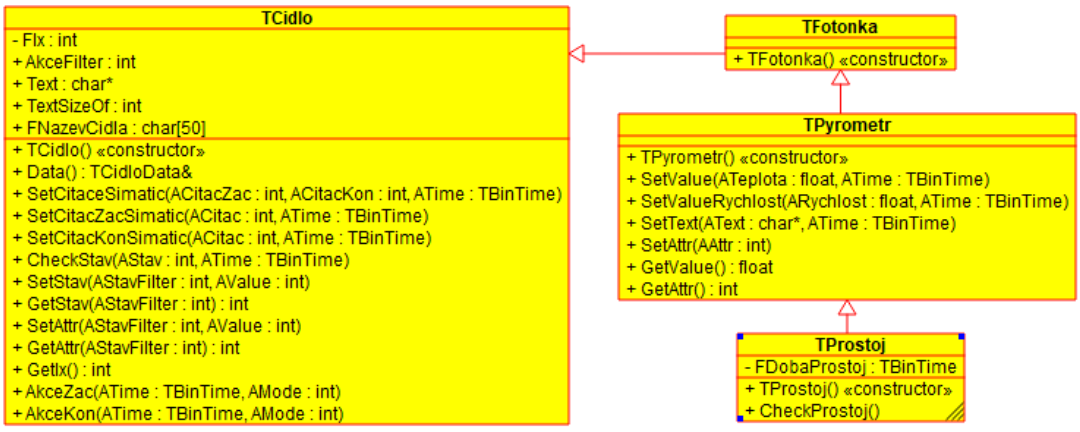
5.2.1 TCidlaData

Struktura *TCidlaData*(obr.č.11) je tvořena 1000 prvky struktury *TCidloData*. V této struktuře se nacházejí atributy pro zaznamenání počtu náběžných či sestupných hran, čas, kdy k těmto signálům došlo a další pomocné hodnoty týkající se daného čidla. Jako čidlo můžeme využít signál z fotozávory, tlakového čidla, signál střihu, či průjezd indukčním čidlem. Také lze tuto strukturu použít ke sledování chodu stroje, kdy se v pravidelných intervalech zaznamenává činnost zařízení.



Obrázek 11: Třídní diagram struktury TCidlaData

K tvorbě a práci s jednotlivými čidly slouží třídy *TCidlo*, *TFotonka*, *TPyrometr* a *TProstoj*. Tyto třídy se nacházejí v modulu CidlaMgr. Základní třídou je třída *TCidlo*. V té se nachází metody k nastavení jednotlivých atributů čidla. Ostatní třídy byly zděděny z této třídy pro specifické typy čidel. Diagram vazeb mezi jednotlivými třídami je na obr. č.12



Obrázek 12: Diagram tříd TCidlo, TFotonka, TPyrometr a TProstoj

5.2.2 TKusyData

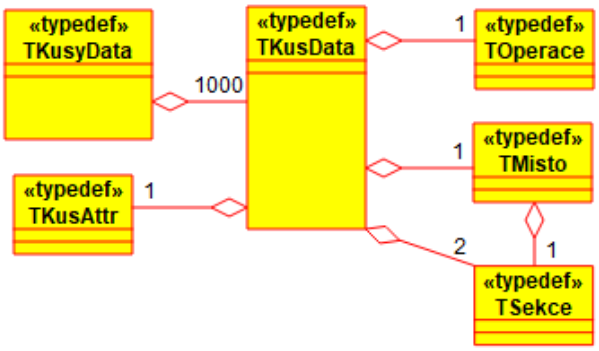
TKusyData je struktura, která je tvořena 1000 prvky struktur *TKusData*. Tyto prvky označují jednotlivé svazky či bedny. Kusy samotné obsahují vlastní atributy, z nichž nejdůležitější je *IdKus*, který jednoznačně identifikuje daný svazek nebo bednu. Tento atribut je také tištěn na výrobní i expediční štítek. Pro zachycení komplexnějších atributů jsem vytvořil struktury *TKusAttr*, *TOperace*, *TMisto* a *TSekce*.

TKusAttr je bitová struktura zachycující příznaky, zda se má svazek načíst z databáze, zda byl již celý zpracován či zda na něm byl vytištěn štítek.

Struktura *TOperace* obsahuje operace, které byly na svazku či bedně provedeny. Může se jednat o sražení hran, rovnání, defektoskopii či demagnetizaci.

Pro přesné zachycení aktuálního umístění svazku či bedny byly vytvořeny struktury *TMisto* a *TSekce*. Ty obsahují pozici a čas, dle kterých lze určit umístění a pořadí kusů v daném místě.

Veškeré vazby mezi strukturami jsou zachyceny na obr. č.13.

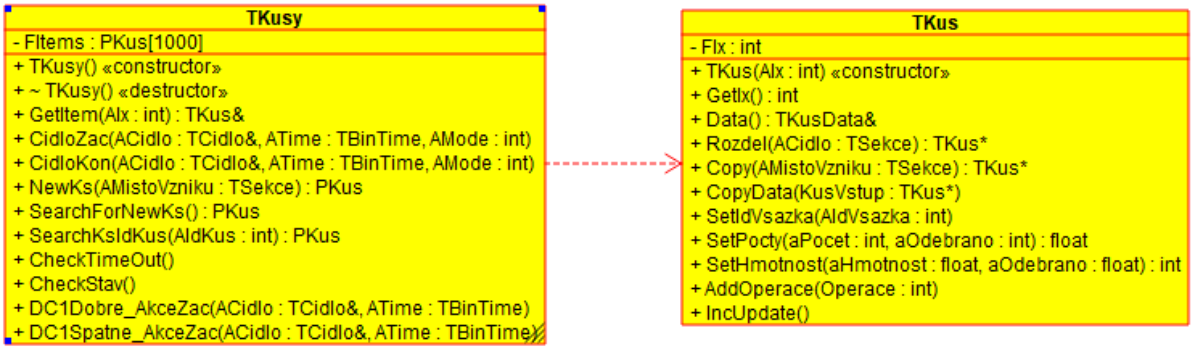


Obrázek 13: Třídní diagram struktury TKusyData

Pro práci se strukturou *TKusyData* byl vytvořen modul KusyMgr. V tomto modulu se nacházejí třídy *TKusy* a *TKus*. Třída *TKus* slouží k manipulaci s jednotlivými položkami ve struktuře. Využívá k tomu metodu *Data()*, která je odkazem na strukturu *TKusyData* v paměťové oblasti daného kusu.

Mezi nejdůležitější metody třídy *TKusy* patří *NewKs()*, *SearchForNewKs()*, *CidloZac()* a *CidloKon()*. Metoda *NewKs()* vrací nový kus a vytvoří jej na požadovaném místě. Stane se tak pouze v případě, je-li k dispozici volné místo ve struktuře. Hledání volného kusu probíhá v metodě *SearchForNewKs()* dle stanovených priorit. Priorita se určuje pozicí, na které se kus nachází. Je-li kus v zóně výběhu či ve skladu, může být použit. Nachází-li se však ještě na některé pozici v rámci stroje, nesmí jej metoda vrátit, aby nedošlo k jeho přepsání.

Metody *CidloZac()* a *CidloKon()* slouží k mapovaní čidel na metody, které obsluhují vykonávání dané akce. Příkladem takové metody je *DC1Dobre_AkceZac()*. V ní je zpracovávána náběžná hrana čidla pro počítání kusů.

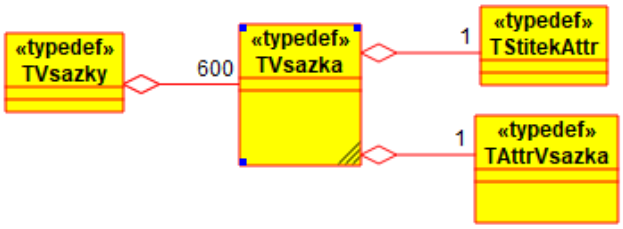


Obrázek 14: Diagram tříd TKusy a TKus

5.2.3 TVsazky

V této struktuře ukládám veškeré informace ze zakázky. Mezi nejdůležitější parametry, patří číslo zakázky, žádaný rozměr, délka, tolerance délky, požadovaný a aktuální počet kusů na zakázku. Jsou zde také uloženy informace, které zákazník vyžaduje na tištěném štítku.

Velikost struktury *TVsazky*(obr. č.15) jsem stanovil na 600 prvků struktury *TVsazka*, tzn. že každý z 10 sledovaných strojů, může využít průměrně 60 zakázek. Takovéto množství dostačuje na naplánování kapacity na několik dnů dopředu, a také zobrazení několikadenní historie. Archiv průběhu výroby je umístěn v databázi.



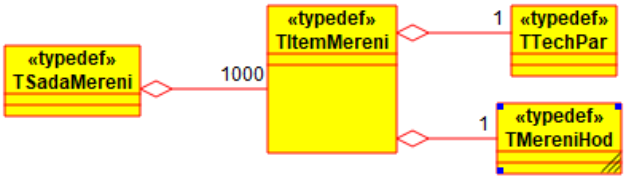
Obrázek 15: Třídní diagram struktury TVsazky

Pro tuto strukturu není vytvořena obslužná třída, pouze modul, obsahující metody pro vytvoření a práci s jednotlivými zakázkami.

5.2.4 TSadaMereni

Aby bylo možné evidovat jednotlivá měření tyčí, vytvořil jsem strukturu o 1000 položkách. V této struktuře jsou zaznamenány podrobné informace o naměřených hodnotách jako jsou např. délka, uhel hrany, šířka hrany apod. Také je zde zachyceno aktuální nastavení stroje v okamžiku měření, aby bylo možné analyzovat případné problémy. Třídní diagram struktury je na obr. č.16

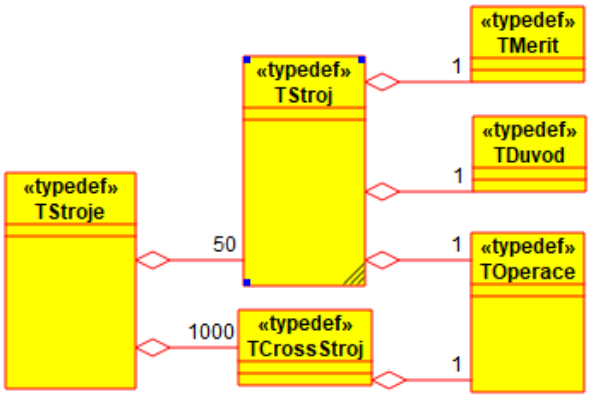
Přístup do této struktury je opět prostřednictvím metod, které přistupují do sdílené paměti.



Obrázek 16: Třídní diagram struktury TSadaMereni

5.2.5 TStroje

Struktura *TStroje* obsahuje aktuální přehled informací o zařízení. Je zde uvedeno číslo vstupního balíku i výstupní bedny, číslo aktuální i následující zakázky a také informace o prostojovosti stroje. Na základě změny některých ze sledovaných atributů se generují záznamy do pomocné struktury *TCrossStroj*, která slouží jako vazební tabulka. Z prvků této struktury, lze následně sestavit tzv. směnové hlášení. Toto hlášení slouží jako protokol toho, co bylo na dané směně vykonáno.



Obrázek 17: Třídní diagram struktury TStroje

Obslužné metody pro tyto struktury jsou definovány v samostatném modulu.

5.2.6 Třída TSimatMgr

Třída *TSimatMgr* je základní třídou, sloužící ke zpracování dat z PLC. Pomocí systému mailboxů odchytává zprávy z procesu ComDC1 a zapisuje je do pomocných čidel. Pokud jsou na tato čidla definované události, zajistí se vyhodnocení a uložení dat do sdílené paměti. Čidla mohou sloužit také pro uložení aktuálního stavu stroje, který je následně zobrazen v klientské aplikaci.

Zápis do PLC probíhá obdobným způsobem. Do předem definovaných míst v databloku jsou zapsány aktuální stavy čidel a dat ze sdílené paměti. Datablok je zapsán do mailboxu, který je procesem ComDC1 postoupen do PLC.

Princip odchytávání zpráv je založen a událostním mechanismu. V metodě *HandleEvent()*, jsou zachyceny zprávy. Ty jsou dvojího typu. Zprávy ze SIMATICu a zprávy periodické.

Periodické zprávy umožňují vykonávat akce v přesně definované periodě. V této práci jsem této skutečnosti využil k získání hodinového výkonu linky zápisem počtu střižených kůsu v minutových intervalech.

Zprávy ze SIMATICu jsou vyhodnocovány samostatnými metodami. Na výpisu č.4 je zobrazena metoda *RecDC1DB21()*, která slouží k zápisu do databloku. Dle struktury databloku jsou do jednotlivých pozic zapsána data z čidel a nebo přímo ze struktur sdílené paměti.

```
void TSimaticMgr::RecDC1DB21(TSimaticDB &Sdb) {
TSimaticRWDB RWDB(IdSimaticDC1, 21, 0);
int i,i2 = 0,IxV=-1;
IxV = SearchIdVsazka(Stroj(stDC1).IdVsazkaAktualni);
if (IxV>=0){
i = DC1Stop.GetStav(0x10) > 0 ? 1 : 0 ;
i2 = DC1Stop.GetStav(C_StopOpravnyMod) > 0 ? 1 : 0 ;
RWDB.Setb(2,0,i);//stop freza
RWDB.Setb(2,1,i2);//stop pohanec
RWDB.Setb(2,7,DC1LaserOnOff.GetStav(0x01) > 0 ? 1 : 0);
RWDB.Setb(3,0,DC1LaserStay.GetStav(0x01) > 0 ? 1 : 0);
RWDB.Setb(3,1,DC1LaserZarovnani.GetStav(0x01) > 0 ? 1 : 0);
RWDB.Setb(3,2,DC1Stop.GetStav(C_StopSerizovani) > 0 ? 1 : 0 );
RWDB.SetI(4, DC1LaserIndex.Data().CitacKonSimatic);
RWDB.SetI(8, DC1LaserPocetDoZast.GetValue());
RWDB.SetI(10, Vsazka(IxV).KusuDoBedny);
RWDB.SetD(20, Stroj(stDC1).IdKusVystup1);
RWDB.SetD(24, Stroj(stDC1).IdKusSroty2);
RWDB.SetD(28, DC1LaserCasStabil.GetValue());
RWDB.SetD(32, DC1LaserCasMereni.GetValue());
RWDB.SetD(36, DC1LaserCasUvolneni.GetValue());
RWDB.SetD(40, DC1LaserCasZpet.GetValue());
RWDB.SetD(44, DC1LaserCasZvednuti.GetValue());
RWDB.SetD(48, DC1LaserCasOfukZap.GetValue());
RWDB.SetD(52, DC1LaserCasOfukDoba.GetValue());
RWDB.SetD(56, DC1LaserCasZarovnani.GetValue());
};
SendDataBlokDC1(RWDB);
}
```

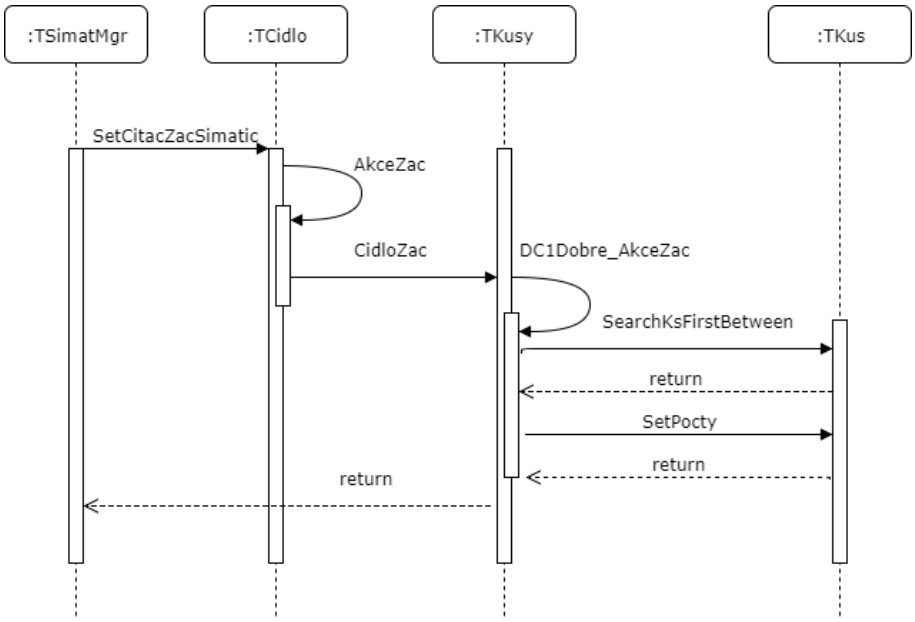
Výpis 4: Metoda RecDC1DB21()

Pro čtecí databloky jsem implementoval dvě metody. Metoda *RecDC1DB18()* (výpis č.5)čte aktuální informace ze stroje, *RecDC1DB24()* zpracovává údaje z laserového měření.

```
void TSimaticMgr::RecDC1DB18(TSimaticDB &Sdb)
{
DC1Strih.SetCitacZacSimatic(Sdb.GetD(2),Sdb.DateTime);
DC1DobreStrih.SetCitacZacSimatic(Sdb.GetD(6),Sdb.DateTime);
DC1Dobre.SetCitacZacSimatic(Sdb.GetD(26),Sdb.DateTime);
DC1Spatne.SetCitacZacSimatic(Sdb.GetD(10),Sdb.DateTime);
DC1TycDelka.SetValue(Sdb.GetI(60));
DC1CidlaChod.SetStav(0x0000000040,Sdb.Getb(50,1)); // Feed sensor
...
}
```

Výpis 5: Metoda RecDC1DB18()

Jakmile dojde k přečtení aktuální hodnoty z databloku a jejímu zápisu do příslušného čidla, může dojít k volání obslužné metody. Tyto obslužné metody se nacházejí ve třídě *TKusy* a slouží k přímému zápisu do sdílené paměti. Na obrázku č.18 je diagram volání metody *SetCitacZacSimatic()* u čidla *DC1Dobre()*. Tato metoda má za úkol navýšit počty u daného kusu.



Obrázek 18: Sekvenční diagram volání ze třídy TSimatMgr

Akce na ostatních čidlech jsou prováděny obdobným způsobem.

Proces SimatMgr je kritickým procesem pro sledování. Jakmile dojde v tomto procesu k chybě, jsou na příslušná místa odeslány informativní SMS, aby došlo k rychlé nápravě.

5.2.7 Procesy AppSrv a ComAppSrv

Ke komunikaci s klientem jsem vytvořil dva procesy: AppSrv a ComAppSrv.

Proces ComAppSrv slouží jako prostředník mezi aplikačním serverem a řídícím serverem. Přenáší data po protokolu TCP/IP. Součástí inicializačních parametrů je IP adresa hostitele a port, po kterém budou procesy komunikovat. Samotná zpráva obsahuje speciální STX a ETX konstanty, aby bylo možné poznat začátek a konec zprávy. Ve čtecí metodě *ReadPort1ToRecMbx()* se poté kontroluje, zda zpráva tyto znaky obsahuje. Naopak metoda *ReadSendMbxToPort1()* tyto znaky do zprávy dodává. V metodách se využívají systémová volání systému OpenVMS.

K samotnému zpracování dat mezi klientem a řídícím serverem jsem vytvořil třídu *TAppSrv*. Tato třída je v samostatně spustitelném souboru. Je postavena na podobném událostním mechanismu jaký má třída *TSimatMgr*. Na základě událostí přijímá či odesílá data na aplikační server.

Základními funkcemi jsou *ZpracujMsgX()* a *SendOneY()*, kde X značí název funkce volané z klienta a Y strukturu ve sdílené paměti.

Ve výpisu č.6 je zobrazena metoda *SendOnePracoviste()*. V této metodě se vytvoří objekt zprávy. Následně se do něj přiřadí jednotlivé parametry. Poté je nastavena hlavička, ve které se nachází identifikátor objektů sdílené paměti. Nakonec je zpráva zapsána do mailboxu, který vyčítá již zmíněný proces ComApSrv. Tento princip je shodný pro všechny objekty sdílené paměti.

```
void TAppSrv::SendOnePracoviste(int Ix)
{
TAppSrvMsg Msg(0);int I=1;
 if (Prac(Ix).Poradi <= 0) return;
 Msg.SetCount(0);
 Msg.SetI(I++, Ix);
 Msg.SetI(I++, Prac(Ix).UpdateCounter);
 Msg.SetI(I++, Prac(Ix).Poradi);
 Msg.SetI(I++, Prac(Ix).IdSmena);
 Msg.SetI(I++, Prac(Ix).IdUzivatel);
 Msg.SetT(I++, Prac(Ix).PrihlasenOd);
 Msg.SetS(I++, Prac(Ix).NazevPracoviste);
 Msg.SetS(I++, Prac(Ix).CeleJmeno);

 if (!MbxAppSrvSend) return;
 Msg.SetHlavicka(msgPracoviste, 0, GetTim());
 Msg.WriteWaitMbx(MbxAppSrvSend);
}
```

Výpis 6: Metoda SendOnePracoviste()

5.3 Archivace dat

Data určená získaná z dělícího zařízení je potřeba ukládat. Nejefektivnějším způsobem je využití databázových systémů. V Třineckých železárnách je použit databázový systém Oracle RDB. Tento systém běží pod operačním systémem OpenVMS firmy Hewlett-Packard.

K systému se přistupuje jednak prostřednictvím ODBC ovladače, a také přímo ze systému OpenVMS skrze SQL services. Základním dotazovacím jazykem je SQL(Structured Query Language).

5.3.1 ODBC - Open Database Connectivity

ODBC je specifikací pro databázové API [9]. Toto API je nezávislé na jakémkoliv SŘDB nebo operačním systému. Je také jazykově nezávislé. ODBC API je založeno na CLI specifikaci od Open Group a ISO/IEC. Funkce v těchto API jsou implementována vývojáři pro dané SŘDB.

5.3.2 Procesy Archiv a Loader

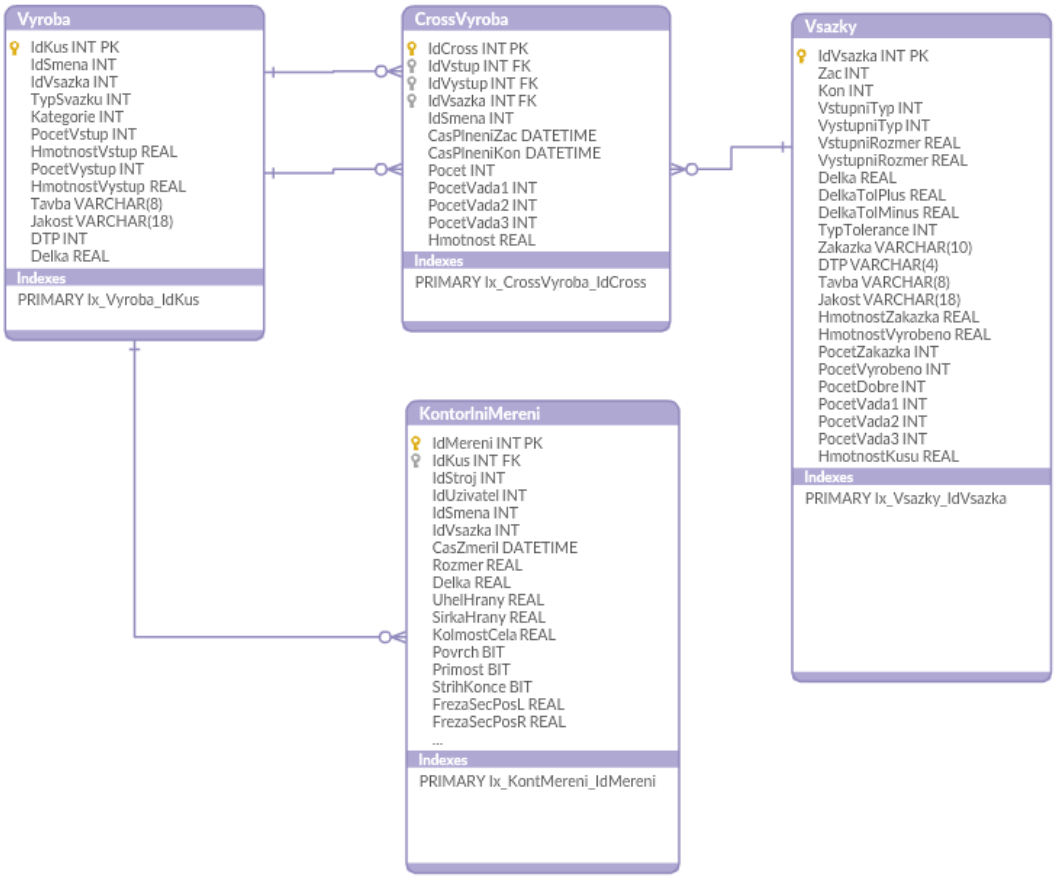
Na řídícím serveru jsem pro archivaci dat vytvořil proces Archiv. Tento proces v předem definovaných intervalech sleduje změny dat ve strukturách sdílené paměti. Pokud se v datech udála změna, provede se uložení do databáze. Dotazy jsou ukládány v samostatných SQM modulech, které se linkují k procesu během jeho kompilace. Po každé úpravě dotazu je nutné proces znovu překompilovat a spustit.

Načítání informací o vstupních svazcích je prováděno v procesu Loader. Tento proces každou sekundu kontroluje, zda nedošlo k požadavku na dočtení informací.

5.3.3 Schéma databáze

Pro potřeby archivace bylo vytvořeno schéma tabulek, které slouží uložení důležitých parametrů.

1. VYROBA - tabulka, která zaznamenává každý kus, se kterým se na stroji pracuje. Jedná se o vstupní svazky, a také o výstupní bedny.
2. VSAZKY - tabulka, obsahující veškeré informace o zakázkách.
3. CROSSVYROBA - vazební tabulka, propojuje informace o jednotlivých kusech a zakázkách, sleduje změny daných parametrů.
4. KONTROLNIMERENI - v této tabulce se nacházejí údaje z měřících zařízení.



Obrázek 19: ER Diagram databáze

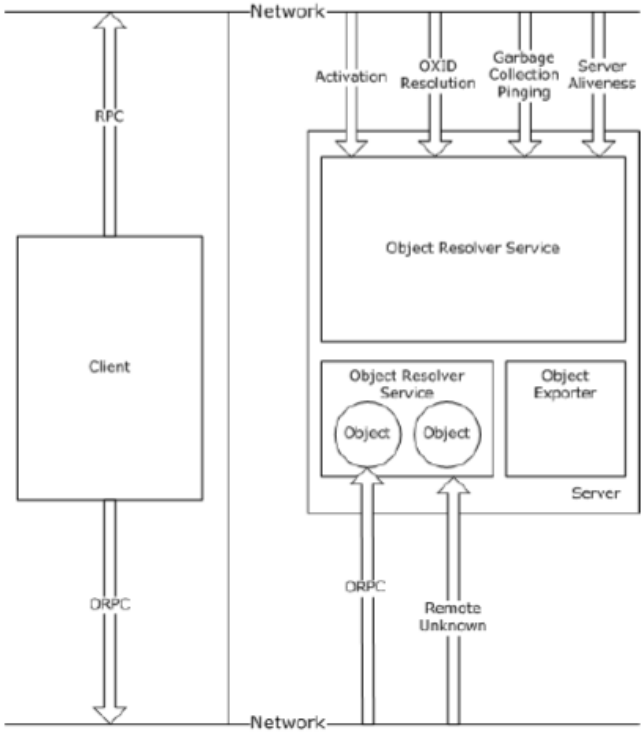
6 Vizualizace dat

V této části popíši komunikační protokol DCOM, který slouží pro přenos dat mezi aplikačním serverem a klientským programem. Dále se zaměřím na implementaci aplikačního serveru. Na závěr rozeberu implementaci klientské aplikace.

6.1 DCOM

DCOM Remote Protokol rozšiřuje Compoment Object Model (COM) o možnost tvorby,aktivace objektů a o zprávu referencí na objekty, jejich životnost a dotazy na rozhraní. DCOM protokol je postaven na RPC protokolu a spoléhá se na autentizaci, autorizaci a schopnostech zachovat celistvost zpráv.

Aplikace vyšších vrstev využívají DCOM klienta k získání reference na objekt a následně na něj uskutečňují ORPC volání. DCOM klient užívá RPC protokol ke komunikaci s objektovým serverem. Toto chování je znázorněno na obrázku č.20



Obrázek 20: Pohled na protokol DCOM (obr. převzat z [3])

6.1.1 Aktivace

Aktivace je základní pojem používaný k popisu tvorby nebo nalezení existujícího DCOM objektu nebo třídní továrny. V protokolu DCOM se užívají dva druhy rozhraní k aktivaci objektů: *IActivation* a *IRemoteSCMActivator*.

Na základní úrovni se aktivace objektů skládá z odeslání následujících informací směrem k aktivační službě objektu:

- Třídního identifikátoru CLSID
- Jednoho z více IID
- Volitelně z odkazu na inicializační umístění

CLSID identifikuje třídu objektu, který má být vytvořen, IID rozpoznává rozhraní u nově vzniklého objektu, o který si žádá klient, a případně nastavuje vlastnosti, se kterými bude objekt inicializován po vytvoření. Aktivace vrací odkaz na objekt zpět klientské aplikaci. Tato aplikace může také odesílat nebo přijímat odkazy na objekty v rámi ORPC volání.

6.1.2 Objektové reference

Odkazy na objekty jsou typu OBJREF. Když je typ OBJREF typ zařazen do DCOM protokolu, NDR dá pokyn k zapouzdření této reference do datového proudu jednotky PDU. Tato data obsahují informace pro klienta, dle kterých je vytvořena RPC vazba na objekt. Protokol DCOM na základě těchto dat vrací odkaz na objekt aplikaci.

6.1.3 DCOM v Delphi

V Delphi lze k připojení k serveru DCOM využít komponentu *DCOMConnection*. Na základě vlastností *ComputerName* určíme adresu, *ServerGUID* nebo *ServerName* umožní propojení na registrovaný objekt.

6.2 Aplikační server

Celá vícevrstvá architektura Delphi pro přístup k datům se točí kolem myšlenky tzv. datových paketů. V tomto kontextu představuje datový paket blok přenášený z aplikačního serveru na klienta a naopak. Datový paket je jistou podmnožinou databázové komponenty. Popisuje data v něm obsažená(zpravidla několik záznamů) společně s názvy a typy datových položek. Datový paket také obsahuje omezení, tj. pravidla aplikovaná na data. Omezení se zpravidla nastavují na aplikačním serveru, přičemž server je zasílá klientským aplikacím společně s daty.

Veškerá komunikace mezi klientem a serverem probíhá na bázi výměny datových paketů. Zdrojová komponenta umístěná na serveru řídí přenos několika datových paketů v rámci větší databázové komponenty, s cílem zajistit pro uživatele rychlejší odezvu. Jakmile klient komponentou *ClientDataSet* obdrží datový paket, může upravovat záznamy v něm obsažené. Klient rovněž obdrží i omezení, která se aplikují během veškerých datových úprav.

Po aktualizaci záznamů klientem a jejich odeslání zpět na server se paketu říká rozdílový paket(tzv. delta). Delta pakety zaznamenávají rozdíly mezi původními záznamy a aktualizovanými. Jakmile klient požaduje uložení změn na serveru, posílá mu tyto rozdílové pakety a server

se pokouší uplatnit všechny změny. Vzhledem k možnému připojení více klientů, nemusí být tyto změny provedeny, neboť data již mohla být změněna.

6.3 Tvorba aplikačního serveru v Delphi

V prostředí Delphi je pro tvorbu serveru vytvořena třída *TRemoteDataModule*.K vytvoření vzdáleného datového modulu je připraven průvodce, který umožní nastavit server dle požadavků na způsob spouštění a přístupu ke klientským požadavkům.

Volby pro přístup ke zpracování klientských požadavků jsou následující:

- Single - datový modul přijímá pouze jeden požadavek v každém časovém okamžiku. Protože jsou klientské požadavky serializovány prostřednictvím COM, není třeba ošetřovat problémy s vlákny.
- Apartment - každá instance vzdáleného datového modulu obsluhuje jeden požadavek. Nicméně, DLL může zpracovávat více požadavků na různých vláknech, pokud vytváří více COM objektů. Instanční data jsou chráněna, ale je třeba řešit možné konflikty v globální paměti.
- Free - datový modul může přijmout více souběžných klientských požadavků ve více vláknech. Je třeba ošetřit jak instanční data, tak i globální paměť.

Lze také zvolit způsob spouštění a to následujícími způsoby:

- Internal - tato volba je vyhrazena pro tvorbu vzdáleného datového modulu, který je součástí dynamicky linkované knihovny.
- Single instance - pouze jedna instance datového modulu je vytvořena pro každý spustitelný soubor. Každý klient spouští svou vlastní instanci serveru.
- Multiple instance - je vytvořena pouze jedna instance aplikace pro všechny datové moduly vytvořené pro klienty. Každý modul je dedikován pro jednoho klienta, ale procesní prostor je sdílen.

Po ukončení průvodce je vytvořena třída, která dědí ze třídy *TRemoteDataModule* a implementuje rozhraní *IAppServer*. Výsledek je zobrazen na výpisu č.7.

```
type
 TMyAppServer = class(TRemoteDataModule, IMyAppServer)
 private
  { Private declarations }
 protected
  class procedure UpdateRegistry(Register: Boolean; const ClassID, ProgID:
     string); override;
```

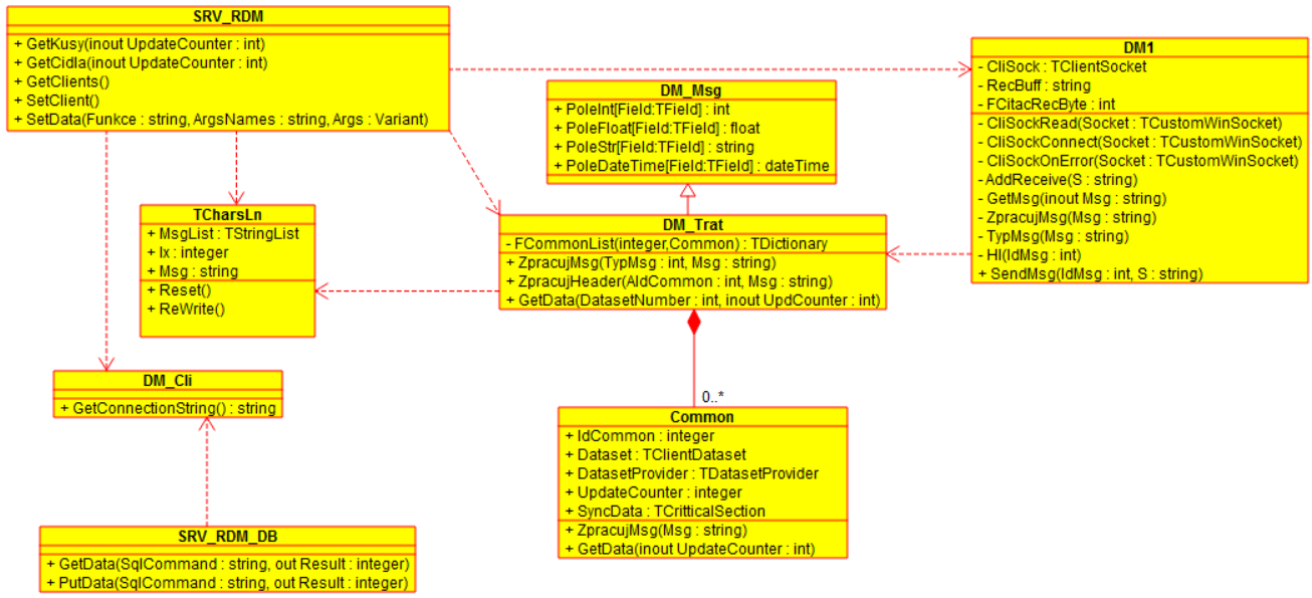
```
 public
  { Public declarations }
end;

initialization
 TComponentFactory.Create(ComServer, TMyAppServer, Class_MyAppServer,
     ciMultiInstance, tmApartment);
```

Výpis 7: Ukázkový aplikační server

6.4 Aplikační server dělícího centra

Pro zpracování dat z řídícího serveru je nutné implementovat na aplikačním serveru třídy, které tyto data zpracují, a budou je dále distribuovat ke klientům. K tomuto účelu jsem implementoval několik datových modulů, které se starají o příjem dat, zpracování a postoupení. Na obr. č.21 je zobrazen diagram tříd aplikačního serveru.



Obrázek 21: Třídní diagram aplikačního serveru

Na diagramu jsou dvě třídy remote data modulu. Jedna slouží ke zpracování požadavků na řídící server, druhá se využívá ke komunikaci s databázovým serverem. Obě třídy komunikují skrze samostatný DCOM objekt. Dále je zde několik tříd dědících z třídy *TDataModule* a pomocné třídy sloužící k analýze řetězců a ukládání dat. Server má také třídu pro vizuální formulář, ten však není využit.

6.4.1 Třída TVF_RDM_Trat

Základní třída remote data modulu, která se stará o komunikaci s klienty ze strany požadavků na řídící server a PLC. Obsahuje metody pro odesílání datasetů na vyžádání a volání funkcí implementovaných na řídícím serveru. Aby bylo omezeno množství odesílaných dat, je v každé metodě parametr poslední aktualizace dat. Ten zajistí, že se na klienta dostanou pouze nová či změněná data, nikoliv vše. U velkých datových struktur je takto ušetřeno několik MB za sekundu.

Příklad jedné z metod je na výpisu č.8

```
function TVF_RDM_Trat_R.GetWatchDogs(var ALastUpdateCounter: Integer):
    OleVariant;
var PomUpdateCounter: integer;
begin
   DsData.Close;
   DsData.Filter := '';
   DsData.IndexFieldNames := '';
   DsData.Filtered := ALastUpdateCounter >= 0;
   PomUpdateCounter := DM_Trat.UpdateCounterWatchDogs;
   DsData.Filter := Format('UpdateCounter>%d', [ALastUpdateCounter]);
   DsData.Data := DM_Trat.DataWatchDogs;
   DsData.First;
   ALastUpdateCounter := PomUpdateCounter;
   Result := Provider_dsData.Data;;
   DsData.Close;
```

Výpis 8: Metoda GetWatchDogs()

V této třídě je také implementována metoda pro zápis dat z klientské strany do serverových struktur. Metoda má název *SetData()* a obsahuje kontrolní mechanismy a převodní funkce, aby se na řídící server dostaly hodnoty ve správném tvaru. Metoda porovná počet zadaných parametrů s počtem vložených hodnot. Poté je odešle na řídící server metodou *SendMsg()*. Tato metoda přidá do zprávy počáteční a ukončovací znaky, identifikátor a odešle zprávu přes TCP/IP protokol, kde jej odchytí proces *ComAppSrv*.

6.4.2 Třída TDM1

Ve třídě *TDM1* jsou implementovány metody pro socketovou komunikaci. Jedná se o metody *CliSockConnect()*, *CliSockOnError()* a *CliSockRead()*. V metodě *CliSockRead()* se v pravidelných intervalech kontroluje příjem zpráv z řídícího serveru. Tyto zprávy jsou následně parsovány medodami *ZpracujX()*. Volba správné metody závisí na hlavičce přijaté zprávy. Interval kontroly je nastaven v časovači třídy *TDelayer*.

Důležité při zpracování je celistvost zpráv. Proto jsou ve zprávě počáteční a koncové značky, pomocí kterých se dá zjistit, zda je zpráva úplná či nikoli. Pro tento účel je vytvořena funkce *GetMsg()*.

Funkce postupně prochází řetězec a hledá počáteční znak zprávy. Po jeho nalezení hledá konec. Pokud jsou oba znaky nalezeny, je z bufferu zkopírována zpráva do vstup-výstupního parametru *Msg*.

Jakmile je zpráva složena, pošle se do datového modulu, který ji zpracuje. Implementovány jsou dva hlavní typy zpráv. Hlavičková zpráva, obsahující názvy, typy a velikosti datových typů a datová zpráva. V datové zprávě jsou názvy atributů a jejich hodnoty.

6.4.3 Třída TCharsLn

Třída *TCharsLn* slouží ke zpracovaní zprávy přicházející ze serveru. Obsahuje metody pro lepší práci s řetězci.

6.4.4 Třída TDM_Trat

V této třídě dochází ke zpracovaní jednotlivých zpráv a jejich převod na data. Obsahuje kolekci objektů třídy *Common*, které nesou informace o jednotlivých strukturách z řídícího serveru. Na základě metody *ZpracujHeader()* rozšiřuje kolekci o nové struktury. Metoda *ZpracujMsg()* tyto struktury plní.

Pro přístup k datům je implementována metoda *GetData()*. Ta na základě jednoznačného čísla identifikuje danou strukturu a vrací její obsah na klientskou stanici.

6.4.5 Třída Common

Veškeré práce nad daty se provádí v objektech této třídy. Třída obsahuje vlastní dataset, ve kterém uchovává data přicházející z řídícího serveru. V konstruktoru třídy je vytvořeno schéma datasetu. Toto schéma vznikne rozkódováním zprávy obsahující hlavičková data.

Zápis do tohoto datasetu probíhá v metodě *ZpracujMsg()*. Zde je nejprve vytvořen objekt typu *TCharsLn*. Následně je volána metoda *Enter()* třídy *TCriticalSection*, která zamkne metodu pro dané vlákno. Poté probíhá samotné zpracování zprávy. Na základě datových typů je hodnota převedena a uložena do datasetu. Metodou *Leave()* je následně uvolněn zámek a zpracování končí. Pseudo-kód metody je uveden ve výpisu č.9.

Metodou *GetData()* jsou vraceny data konkrétních struktur. Vstup-výstupní parametr *AUpdateCounter* je přijímán z klienta a určuje poslední záznam, který má klient k dispozici. Dle tohoto parametru se vyhodnotí, kolik objektů se má na klientskou stanici odeslat. Výchozí hodnotou je -1. Tato hodnota značí odeslání všech dostupných objektů.

```
Vstup do chranene sekce
preloz zpravu
najdi spravny radek
```

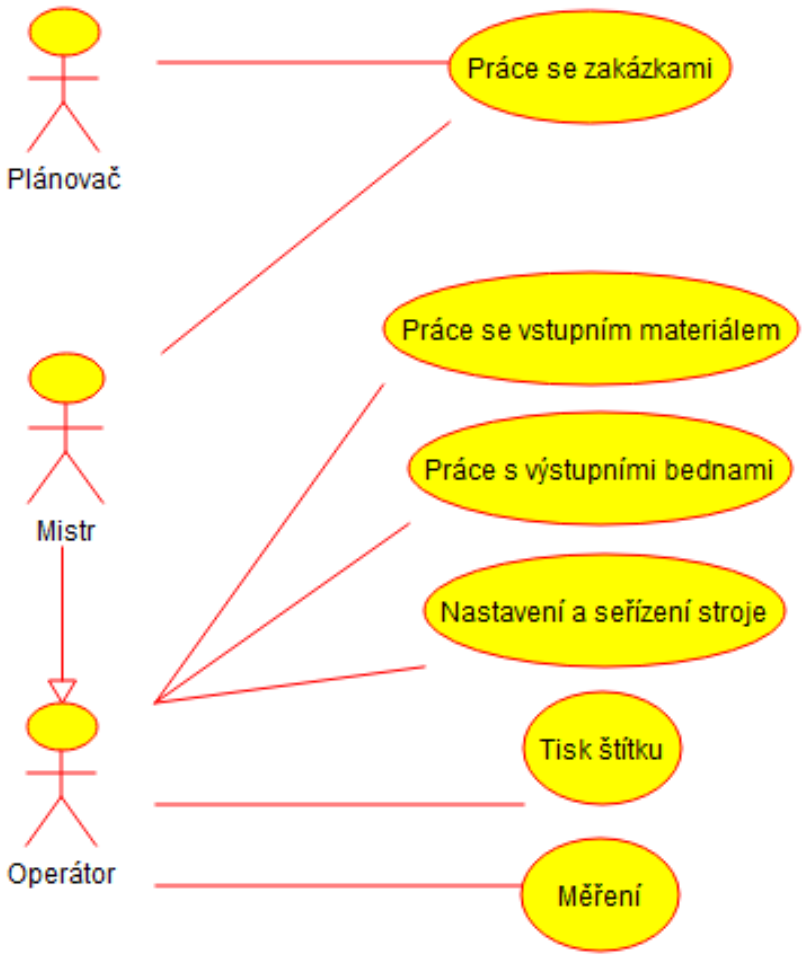
```
for i = 0 to pocet atributu
   pretypuj hodnotu na spravny formatu
   uloz hodnotu
zvys hodnotu updatecounteru
Opust chranenou sekci
```

Výpis 9: Pseudo kod metody ZpracujMsg()

6.5 Návrh klientské aplikace pro sledování toku materiálu

V této části popíši klientskou aplikaci, sloužící ke sledování a ovládání dělícího centra. Aplikace je využívána několika profesemi od plánovačů, přes mistry až po samotné operátory se strojem.

Na obr. č.22 je diagram užití, ve kterém jsou vyznačeny jednotlivé uživatelské role a jejich úkony.



Obrázek 22: Use-case diagram klientské aplikace

Plánovači mají na starost tvorbu zakázek a určení pořadí výroby. Obsluha stroje poté podle těchto parametrů nastavuje stroj, vkládá vhodný materiál a expeduje bedny.

Aplikace byla vytvořena dle následujících požadavků:

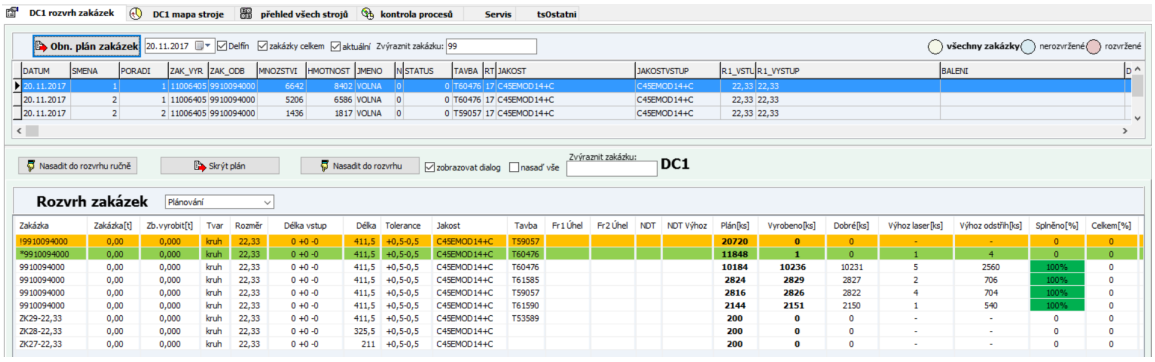
1. Zajistit přístup k datům z plánovacího ERP systému.
2. Umožnit vytvoření plánu výroby s možností doplnění potřebných parametrů.
3. Sledovat plnění zakázky.
4. Zobrazit aktuální stav výrobního zařízení, včetně nastavených parametrů.
5. Načítat čárové kódy vstupních svazků.
6. Komunikovat s měřícími zařízeními pro kontrolu výrobních parametrů.

Dle těchto požadavků vzniklo několik obrazovek.

6.5.1 Plánování

Vzhled plánovací obrazovky byl vytvořen na základě požadavků provozu, aby co nejvíce odpovídal potřebám pro rychlou orientaci v zakázkách. Existují různé pohledy pro plánovače a výrobu, neboť určité parametry jsou důležitější pro obsluhu a naopak. Data jsou periodicky načítána ze serveru, aby reflektovala aktuální situaci.

Pohled pro plánování je na obrázku č.23 Je na něm vidět rozdělení obrazovky na dvě části. V horní části se nachází parametry zakázek, které jsou na vyžádání načítány z plánovacího systému. Ten pracuje s daty ze SAPu a připravuje data do podoby vhodné pro zaplánování. Je na plánovači, v jakém pořadí jsou tyto zakázky navedeny do systému. Každou zakázku, lze také editovat a přiřadit ji výrobní parametry, které se v systému SAP nenacházejí.

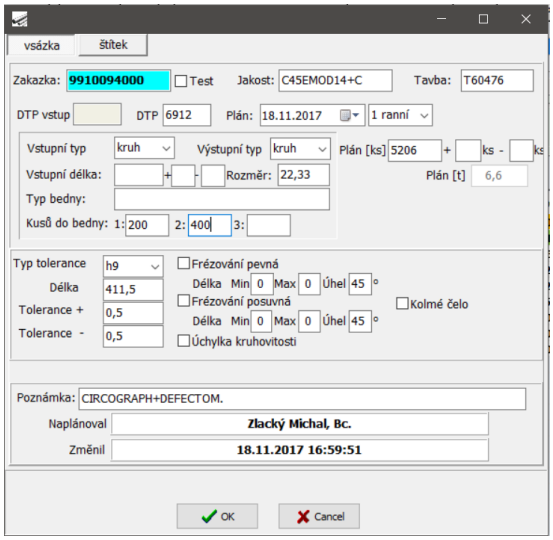


Obrázek 23: Obrazovka plánování dělícího centra

Komunikace s plánovacím systémem je vytvořena v samostatném datovém modulu. Ten prostřednictvím protokolu DCOM volá metodu na vzdáleném serveru. Vrácená data jsou uložena v lokálním datasetu.

Spodní část obrazovky je vyhrazena pro naplánované zakázky. Výrazným prvkem je zde barevné odlišení aktuální a následující zakázky od ostatních. Zakázky, které ještě nebyly zahájeny lze v rámci plánu přesouvat.

V detailním pohledu (obr. č.24) je k zakázce možno přiřadit parametry jako jsou počet kusů do bedny nebo upravit množství na zakázku. K uložení slouží metoda *SetVsazka()*, která sesbírá parametry z jednotlivých editačních polí, naformátuje je do řetězce a odešle na server ke zpracování.



Obrázek 24: Detailní pohled na zakázku

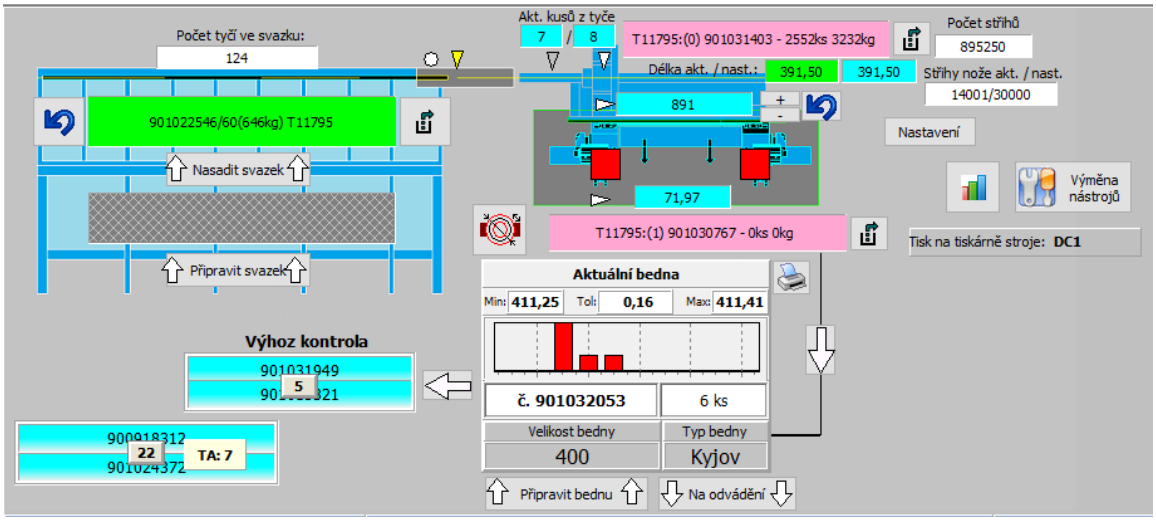
6.5.2 Mapa stroje

Tato záložka je základní obrazovkou pro obsluhu stroje. Nachází se zde veškeré informace o zakázce, nastavení stroje, vlastnostech vstupního materiálu a bednách. V dolní části obrazovky jsou umístěny pole pro přihlášení uživatele ke konkrétní pracovní funkci. Na této záložce dochází každou vteřinu k aktualizaci údajů ze serveru. Také z ní lze nastavovat určité parametry stroje prostřednictvím volání metod na server. Na obr. č.25 je zobrazen výřez této mapy týkající se stroje. Na celkovém pohledu(obr. č.31) se dále nacházejí informace o zakázce, záznamy z automatického a ručního měření, průběh posledních 30-ti měření a aktuálně přihlášení uživatelé.

Pracovní povinnosti uživatele jsou následující:

1. Nasadit vstupní svazek načtením čárového kódu
2. Každou vyrobenou bednu označit výrobním štítek
3. Po splnění požadovaného počtu kusů na zakázku, přepnout na další zakázku.

K těmto účelům jsou v mapě stroje umístěny ovládací prvky, které příslušné akce provádějí.



Obrázek 25: Bližší pohled na mapu stroje

K nasazení vstupního materiálu slouží tlačítko *Připrav vstup*. Po jeho stisknutí se objeví dialog, ve kterém lze zadat číslo svazku. Číslo lze zadat buďto ručně nebo pomocí čtečky čárových kódů. K použití čtečky musí být na počítači nainstalován příslušný ovladač. Data ze čtečky se přijímají a překládají v datovém modulu *TDM_Ctecka*.

K práci s výstupními bednami jsem vytvořil třídu *TFrameBedna*. V této komponentě se zobrazuje číslo bedny, aktuální počet kusů, maximální počet a typ bedny. Také je zde prostřednictvím histogramu vidět, jak se odchyluje požadovaná stříhaná délka v rámi tolerančních mezí. Operátor je tak neustále informován, zda není potřeba seřídit stroj, aby bylo dosaženo co nejlepších výsledků.

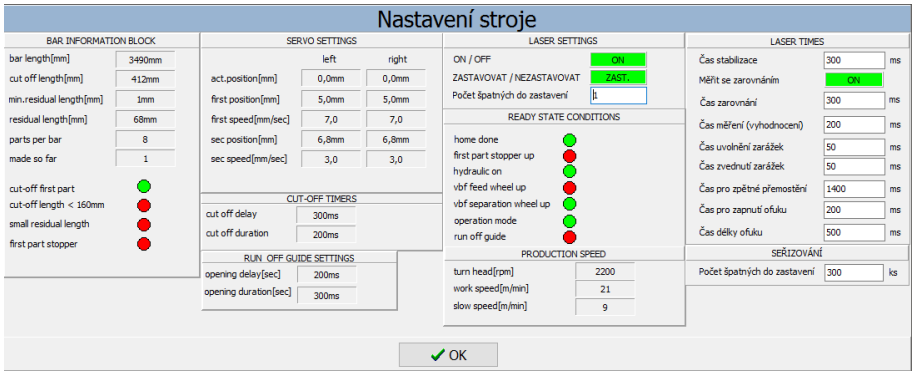
Algoritmus histogramu(výpis č.10) pracuje s hodnotami z laserového měřícího zařízení. Počet hodnot je nastaven na 30. Na základě délkových tolerančních mezí je určen počet binů, na které je histogram rozdělen. Poté se každé z každého měření odečte zakázková délka a tento rozdíl je zařazen do příslušného pole histogramu.

```
numBins:= round((pomTolMax + pomTolMin)*10);
while ((Dm1.dsMereni.RecNo < 30) and (not dm1.dsMereni.Eof)) do
begin
   pomDelta:= DM1.dsMereni.FieldByName('DelkaF').AsFloat - pomDelka;
   for I := -round(numBins/2) to round(numBins/2 - 1) do
   begin
      if ((pomDelta >= I*0.1) and (pomDelta <= (I+1)*0.1))
      then inc(pomPole[I+round(numBins/2)]);
   end;
   DM1.dsMereni.Next;
end;
```

```
Series1.Clear;
for I := -round(numBins/2) to round(numBins/2 - 1) do
begin
   Series1.AddXY(I*0.1,pomPole[I+round(numBins/2)],'');
end;
```

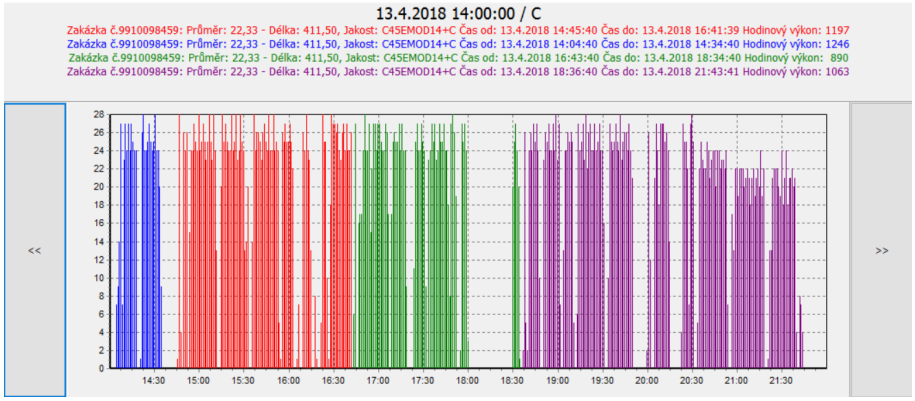
Výpis 10: Algoritmus histogramu měření

Z mapy stroje lze také nastavit některé parametry stroje a laserového měření. Lze tak učinit z dialogu, který je zobrazen po kliknutí na tlačítko nastavení.



Obrázek 26: Dialog nastavení stroje

Posledním důležitým prvkem na mapě je tlačítko grafu. V dialogu spuštěném na akci tohoto tlačítka je zobrazen časový průběh výroby po směnách. V každé směně je uvedena zakázka a hodinový výkon.



Obrázek 27: Graf průběhu výroby na dělícím centru

6.5.3 Kontrola procesů

Na této záložce jsou zobrazeny veškeré běžící procesy pro provoz tažírny oceli. Na základě tzv. watchdogů posílá každý proces informace o tom, zda normálně funguje, případně dodatečná

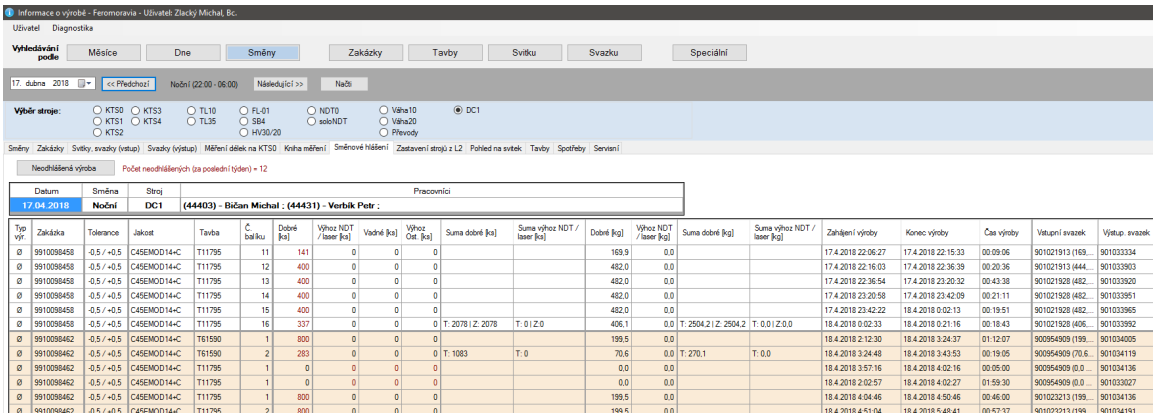
data jakými jsou např. počet tisků, počet přenesených dat atp. Samotná funkčnost procesu je barevně vyznačena. Zelená barva značí normální provoz, je-li proces v červené barvě, znamená to chybu.

Název	PID	Start/Stop	WatchDog	Poslední stav	Kód	Stav - Popis
AppSrv2	124161249	27.3.2018 8:20:10	31.3.2018 9:03:34	31.3.2018 9:03:35	951	GetProcStatus(124161249): Bod2
Archiv	124040836	17.3.2018 15:41:34	31.3.2018 9:03:18	31.3.2018 9:03:34	10	Program. cyklus - cekam na udalost.
ComApSrv2	124024435	17.3.2018 15:40:44	31.3.2018 9:03:35	31.3.2018 9:03:35	600	Odesilam data do TCP/IP portu.
ComApSrv3	124024436	17.3.2018 15:40:44	31.3.2018 6:34:05	31.3.2018 6:34:05	200	Cekam na spojeni - client: "vi-fer-srv2"
ComDC1	124040813	17.3.2018 15:40:43	31.3.2018 9:03:35	31.3.2018 9:03:35	220	Cyklus - cekam 1 sekundu.
ComKTS0	124024423	17.3.2018 15:40:43	29.3.2018 21:42:35	31.3.2018 9:02:42	10	Pripojuji se.
ComKTS3	124040810	17.3.2018 15:40:43	31.3.2018 9:03:35	31.3.2018 9:03:35	210	Cteni databloku ze simatiku
ComKTS4	123979371	17.3.2018 15:40:43	31.3.2018 9:03:35	31.3.2018 9:03:35	210	Cteni databloku ze simatiku
ComLODB	124040814	17.3.2018 15:40:43	31.3.2018 9:03:35	31.3.2018 9:03:35	220	Cyklus - cekam 1 sekundu.
ComPrn1	124040815	17.3.2018 15:40:44	31.3.2018 9:03:34	31.3.2018 9:03:34	100	Cekam na data. Provedeno 3972 tisku
ComPrn3	124177178	28.3.2018 11:13:27	31.3.2018 9:03:34	31.3.2018 9:03:34	100	Cekam na data. Provedeno 94 tisku
ComSB4	124040808	17.3.2018 15:40:43	29.3.2018 21:46:57	31.3.2018 9:03:05	10	Pripojuji se.
ComSoloNDT	124040809	17.3.2018 15:40:43	31.3.2018 9:03:34	31.3.2018 9:03:35	220	Cyklus - cekam 1 sekundu.
ComTL35KN	123979372	17.3.2018 15:40:43	29.3.2018 21:42:38	31.3.2018 9:02:53	10	Pripojuji se.
HodVyr	124040838	17.3.2018 15:41:44	31.3.2018 9:03:26	31.3.2018 9:03:35	10	Program. cyklus - cekam na udalost.
Loader	124040830	17.3.2018 15:41:04	31.3.2018 9:03:34	31.3.2018 9:03:35	10	Program. cyklus - cekam na udalost.
Prenos	124040833	17.3.2018 15:41:14	31.3.2018 9:03:29	31.3.2018 9:03:34	10	Program. cyklus - cekam na udalost.
PrenosDel	124040835	28.3.2018 13:11:42	31.3.2018 9:03:20	31.3.2018 9:03:34	10	Program. cyklus - cekam na udalost.
SimatMgr	124148362	26.3.2018 9:26:50	31.3.2018 9:03:34	31.3.2018 9:03:35	10	Program. cyklus - cekam na udalost.

Obrázek 28: Záložka kontroly procesů

6.6 Zobrazení dat o výrobě a chodu stoje

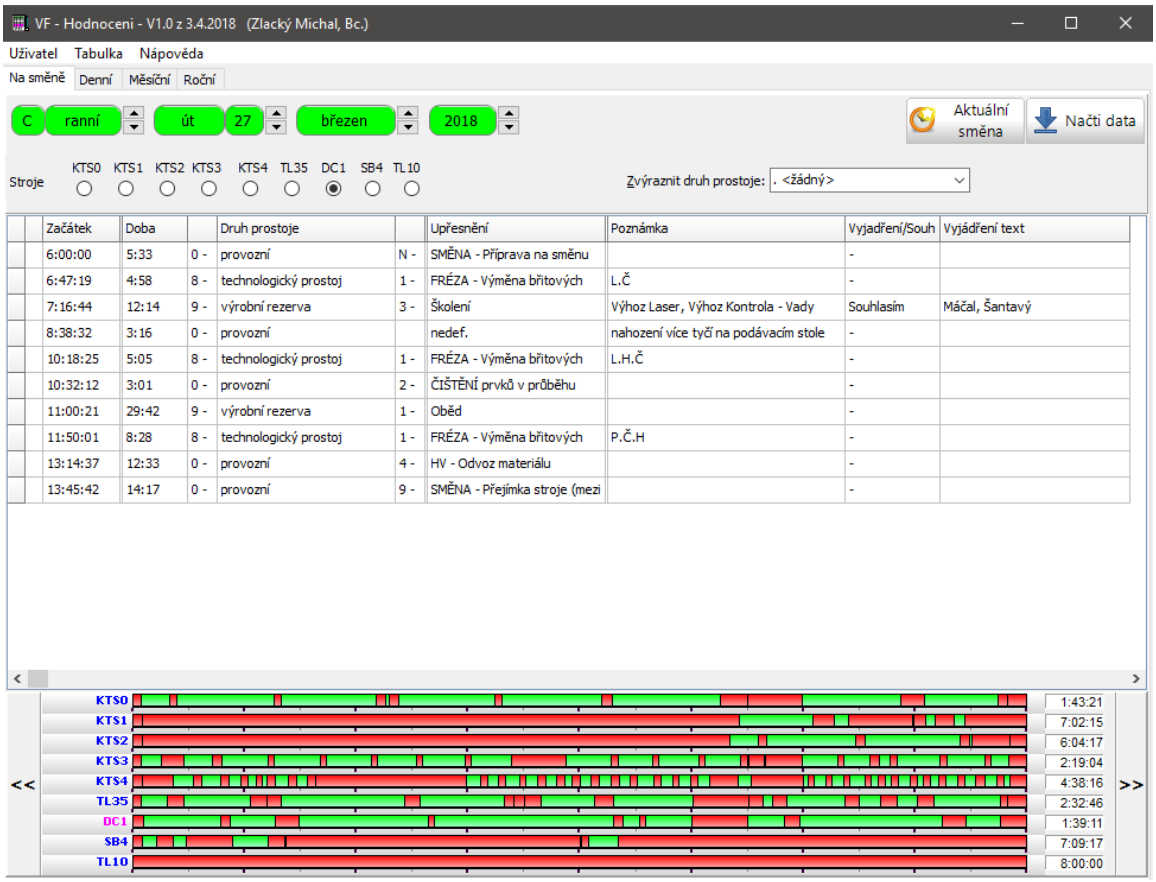
Klientská aplikace vytvořená v této práci slouží výhradně k zobrazení aktuálních dat a naplánovaných zakázek. K zobrazení historie výroby na strojích slouží specializované aplikace vytvořené většinou na míru danému provozu. Pro provoz ve Starém Městě byla vytvořena aplikace VF Info obsahující informace o všech sledovaných výrobních a nevýrobních zařízeních. V aplikaci lze listovat po jednotlivých směnách, dnech či měsících. Také je možné vyhledávat dle různých kritérií. Pohled na tzv. směnové hlášení je na obr. č.29. Na tomto pohledu je vidět celý průběh výroby za jednotlivé směny.



Obrázek 29: Aplikace VF Info

Další důležitá věc u STM je sledování důležitých KPI parametrů. Mezi takovéto ukazatele patří i celková efektivita zařízení (CEZ). Tato efektivita je tvořena z chodu agregátu. Jestliže stroj nepracuje, automaticky mu klesá efektivita. Kromě efektivity lze také zaznamenávat jednotlivé příčiny prostoje zařízení. Třinecké železárny a.s. používají systém pro sledování chodu jednotlivých agregátů. V rámci sledování toku materiálu je zaznamenáván také chod stroje. Prostoj vzniká po určité době nečinnosti stroje. Tento prostoj se zapisuje do speciálních tabulek, na které je navázána aplikace pro jejich zobrazení.

Linka dělícího centra byla zapracována do stávající aplikace sledující prostojovost agregátů na provoze. Jedná se aplikaci VF Hodnocení (obr č.30), v níž lze po směnách editovat jednotlivé prostoje, kategorizovat je a upřesňovat jejich příčiny. Také je zde možné vytvářet denní, měsíční či roční sumární pohledy.



Obrázek 30: Aplikace VF Hodnocení

Mezi další důležité aplikace patří skladové a expediční aplikace. V těch jsou zobrazeny pohyby na skladech vstupního materiálu i expedovaný materiál.

6.7 Budoucí plány

Ke zvýšení efektivity zpracování materiálu připravil provoz na rok 2018 investici ve formě zužitkování materiálu, který byl defektoskopickou linkou vyhodnocen jako nevhodný. Na konec defektoskopické linky bude instalován laser na vypálení QR kódu k jednoznačné identifikaci každé tyče. Na dělící centrum se poté instaluje čtečka těchto kódu.

Postup bude následující:

Defektoskopická linka uloží informace o tyči do databáze ve formě vzdálenosti, ve které se vada nachází a její délky. Po načtení svazku budou dočteny veškeré informace o každé tyči. Tyč je rozdělena na úseky po 2 mm s údajem, zda se v daném segmentu nachází vada. Po načtení tyče čtečkou budou na základě zakázkové délky vyhodnoceny dělené tyče, které se mají ze vstupní tyče vystřihnout. Tímto postupem lze zužitkovat velké množství materiálu, který by byl jinak odvezen k sešrotování.

Na rok 2019 se provoz rozhodl vybudovat novou zušlechťovací linku, jejíž součástí bude úplně nové dělící centrum. Toto centrum bude taktéž začleněno do informačního systému TŽ a.s. Struktury užité v této práci byly koncipovány, aby bylo možné jednoduše integrovat toto zařízení do stávajícího systému.

7 Testování

V rámci vývoje aplikace jsem provedl několik testů rychlosti přenosu a komunikace mezi jednotlivými procesy. Zjišťoval jsem tak možné výkonnostní problémy. Průměrný takt linky je závislý na stříhaném průměru a délce vstupní tyče. V reálném provozu se pohybuje cca 20 střihů za minutu.

Tabulka 2: Test rychlosti komunikace mezi procesy

Procesy	Čas ϕ[s]
PLC → ComDC1	0.6
ComDC1 → SimatMgr	0.1
AppSrv → Aplikační server	1.1
Aplikační server → Klientská aplikace	0.3

Vzhledem k zastavování linky na přesný počet tyčí v bedně je nejdůležitější rychlost komunikace mezi PLC a procesem SimatMgr, ve kterém se vyhodnocování provádí. Dle naměřených časů je tato rychlost dostačující.

7.1 Informace o chybách a zastavení stroje

Do klientské aplikace je integrován systém varovných hlášení, který operátory informuje o různých důvodech zastavení stroje. Ty mohou být z několika příčin. Může se jednat o naplnění bedny na požadovaný počet, dosažení počtu kusů na zakázce, zpracování vstupního svazku nebo také povinnost ručně proměřit veškeré parametry tyče. Nastane-li některá z těchto událostí je do SIMATICu odeslán zastavovací povel, který přeruší střih a posun materiálu. Jakmile je splněna podmínka pro odblokování stroje, může operátor opět vyrábět.

7.2 Zotavení z chyb

Není-li dostupný aplikační server, je v klientské aplikaci zobrazen dialog s možností dalšího pokusu o spojení či vypnutí aplikace. Bez funkčního připojení k aplikačnímu serveru nelze tisknout expediční štítky, které jsou nezbytné pro jednoznačnou identifikaci každé vyrobené bedny. Po obnovení komunikace jsou dopočítány kusy, které byly během výpadku vyrobeny. Ty jsou vypočítány jako rozdíl aktuálního a předchozího nekonečného čítače.

8 Závěr

Automatizace výroby je v dnešní době velmi významné téma. Velké firmy investují nemalé částky do systémů, jež jim umožňují řídit výrobní zařízení a sledovat tok materiálu. Výhodou těchto systémů je také provázanost s účetními systémy, která snižuje lidskou režii. Dalším plusem je možnost sledovat různé ukazatele výkonnosti, jakým je například celková efektivita zařízení. Lze tak optimalizovat výrobu či snížit náklady.

V této práci jsem vytvořil několik programů. Program ke komunikaci s PLC, další obsluhující zprávu sdílené paměti a také aplikaci pro vizualizaci výrobních dat. Tyto programy slouží ke sledování toku materiálu na Dělícím centru v závodě Ferromoravia a.s., který je součástí společnosti TŽ a.s. jako provoz VF. Toto dělící centrum, izraelské firmy VIDEX, bylo instalováno v roce 2016. Jedná se o impaktní řezačku, sloužící k dělení taženého materiálu na servisní tyče požadované délky. Součástí instalace bylo také toto sledování. Dosud bylo na tomto centru nastříháno několik set tun vstupního materiálu na konkrétní zakázkové délky.

Zařízení je řízeno pomocí programovatelného ovladače. PLC, které bylo použito, je od firmy Simatic. Jedná se o model S7-300, který posílá údaje ze stroje ve formě datablоků prostřednictvím protokolu S7. Protokol samotný a jeho implementace jsou popsány v kapitole Sběr dat této práce.

Ke zpracování výrobních dat jsem implementoval několik obslužných programů v jazyce C++. Programy komunikují se strukturami, které jsou kvůli rychlému přístupu k datům namapovány ve sdílené paměti systému Open VMS. V těchto strukturách lze nalézt informace o zakázkách, měření či výrobku. Všechny programy běží na podnikovém serveru Třineckých železáren a.s. Tento server komunikuje s aplikačním serverem v závodě ve Starém Městě. Pro trvalé uložení výrobních informací jsem vytvořil schéma tabulek, do kterých přes archivační proces zapisuji data ze sdílené paměti.

Jak aplikační server tak i klientská část, která se stará o vizualizaci a interakci uživatelů, jsou naprogramovány v jazyce Object Pascal. Klient umožňuje plánovat zakázky z ERP systému TŽ a.s., zobrazovat aktuální nastavení výrobního zařízení, a také informovat operátory o aktuálním průběhu výroby. Aplikace je zařazena do stávajícího informačního systému Třineckých Železáren a.s., ze kterého ji lze spustit. Dle nastavených oprávnění lze poté provádět výše zmíněné úkony. V tomto systému jsou také dodatečné aplikace, založené na datech ze sledování výroby.

V rámci lepšího využití neshodného materiálu byla na rok 2018 naplánována investice k identifikaci a čtení jednotlivých tyčí, které jsou následně stříhány na dělícím centru. Také bylo rozhodnuto o výstavbě nového, plně automatizovaného řezacího centra.

Provoz VF je v rámci skupiny Třineckých železáren a.s. inovátorem v technologické oblasti a dle mého názoru je plně připraven na nadcházející éru průmyslu 4.0.

Literatura

[1] Marco Cantú *Myslíme v jazyku Delphi 7 knihovna zkušeného programátora*, GRADA, 2003

[2] Steve Teixera a Xavier Pacheco *Borland Delphi - průvodce vývojáře*, COMPUTER PRESS, 2002

[3] [MS-DCOM]: Distributed Component Object Model (DCOM) remote protocol.

[4] SNAP7

http://snap7.sourceforge.net

[5] Protocol S7

http://gmiru.com/article/s7comm/

[6] libnodave

https://sourceforge.net/projects/libnodave/

[7] Siemens S7-PLCSIM

http://www.w3.siemens.com/mcms/simatic-controler-software/step7

[8] CreateFileMapping

https://msdn.microsoft.com/en-us/library/aa366537.aspx

[9] What Is ODBC?

https://docs.microsoft.com/en-us/sql/odbc/reference/what-is-odbc

[10] Mezi-procesová komunikace na systému OpenVMS

http://h41379.www4.hpe.com/openvms/journal/v9/mailboxes.pdf

[11] RFC1006

https://tools.ietf.org/html/rfc1006

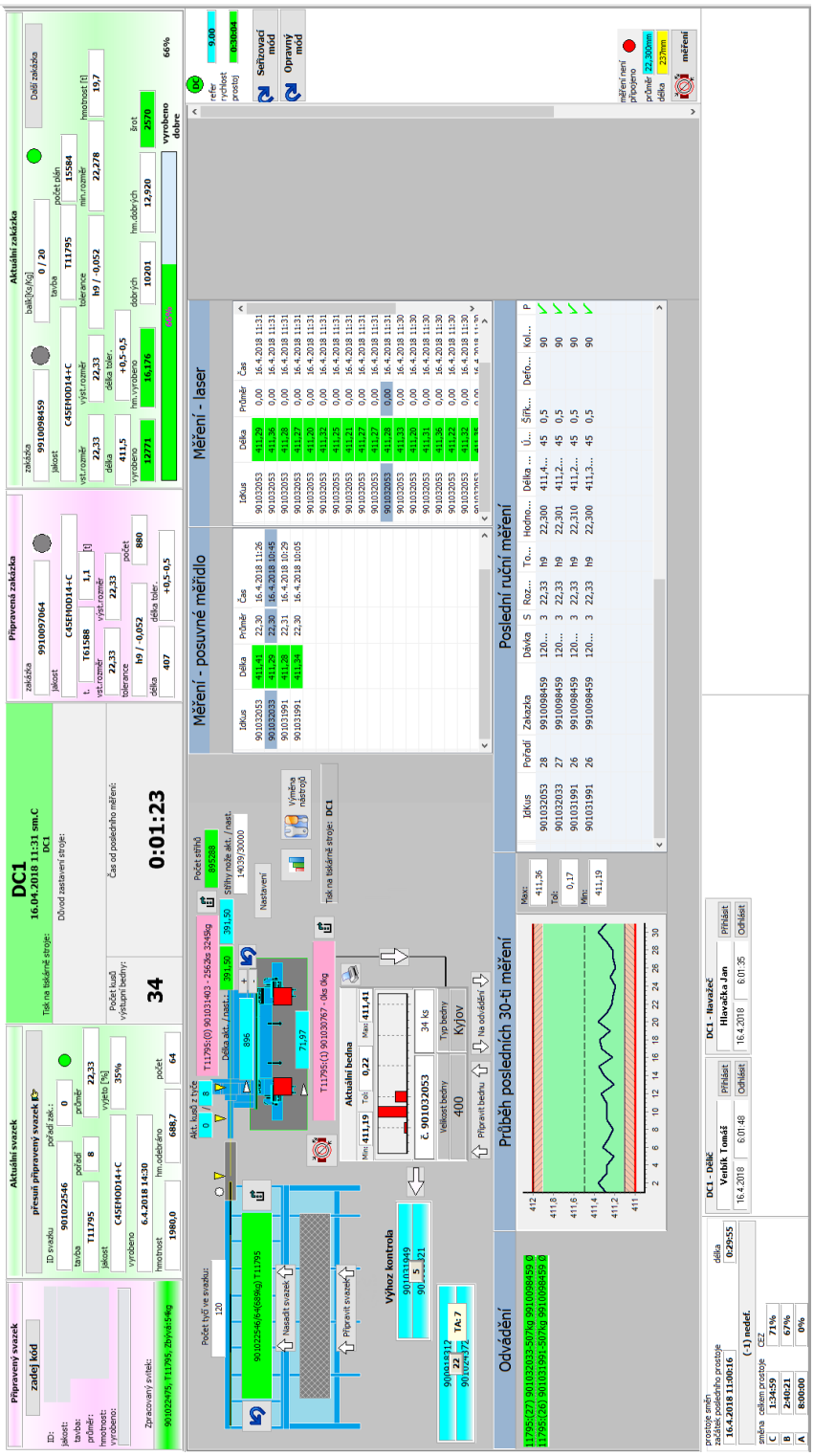
[12] RFC2126

https://tools.ietf.org/html/rfc2126

[13] RFC905

https://tools.ietf.org/html/rfc905

A Mapa stroje



Obrázek 31: Úplná mapa stroje